NAME

fmtmsg – display a message on stderr or system console

SYNOPSIS

#include <fmtmsg.h>

int fmtmsg(long *classification***, const char** **label***, int** *severity***, const char** **text***,**
const char **action***, const char** **tag***);**

MT-LEVEL

Safe

DESCRIPTION

Based on a message's classification component, **fmtmsg( )** writes a formatted message to **stderr**, to the console, or to both.

fmtmsg( ) can be used instead of the traditional **printf**(3S) interface to display messages to **stderr**. **fmtmsg( ),** in conjunction with **gettxt( ),** provides a simple interface for producing language-independent applications.

A formatted message consists of up to five standard components as defined below. The component, *classification*, is not part of the standard message displayed to the user, but rather defines the source of the message and directs the display of the formatted message.

classification
: Contains identifiers from the following groups of major classifications and subclassifications. Any one identifier from a subclass may be used in combination by ORing the values together with a single identifier from a different subclass. Two or more identifiers from the same subclass should not be used together, with the exception of identifiers from the display subclass. (Both display subclass identifiers may be used so that messages can be displayed to both **stderr** and the system console).

- "Major classifications" identify the source of the condition. Identifiers are: **MM_HARD** (hardware), **MM_SOFT** (software), and **MM_FIRM** (firmware).
- "Message source subclassifications" identify the type of software in which the problem is spotted. Identifiers are: **MM_APPL** (application), **MM_UTIL** (utility), and **MM_OPSYS** (operating system).
- "Display subclassifications" indicate where the message is to be displayed. Identifiers are: **MM_PRINT** to display the message on the standard error stream, **MM_CONSOLE** to display the message on the system console. Neither, either, or both identifiers may be used.
- "Status subclassifications" indicate whether the application will recover from the condition. Identifiers are: **MM_RECOVER** (recoverable) and **MM_NRECOV** (non-recoverable).
- An additional identifier, **MM_NULLMC**, indicates that no classification component is supplied for the message.

label
: Identifies the source of the message. The format of this component is two fields separated by a colon. The first field is up to 10 characters long; the second is up to 14 characters. Suggested usage is that *label* identifies the package in which the application resides as well as the program or application name. For example, the

label **UX:cat** indicates the UNIX System V package and the **cat** application.

severity
: Indicates the seriousness of the condition. Identifiers for the standard levels of *severity* are:

- **MM_HALT** indicates that the application has encountered a severe fault and is halting. Produces the print string **HALT**.
- **MM_ERROR** indicates that the application has detected a fault. Produces the print string **ERROR**.
- **MM_WARNING** indicates a condition out of the ordinary that might be a problem and should be watched. Produces the print string **WARNING**.
- **MM_INFO** provides information about a condition that is not in error. Produces the print string **INFO**.
- **MM_NOSEV** indicates that no severity level is supplied for the message.

Other severity levels may be added by using the **addseverity( )** routine.

text Describes the condition that produced the message. The *text* string is not limited to a specific size.

action Describes the first step to be taken in the error recovery process. **fmtmsg( )** precedes each action string with the prefix: **TO**FIX**:.** The *action* string is not limited to a specific size.

tag An identifier which references on-line documentation for the message. Suggested usage is that *tag* includes the *label* and a unique identifying number. A sample *tag* is **UX:cat:146**.

Environment Variables

There are two environment variables that control the behavior of **fmtmsg( ) : MSGVERB** and **SEV_LEVEL**.

MSGVERB tells **fmtmsg( )** which message components it is to select when writing messages to **stderr**. The value of **MSGVERB** is a colon-separated list of optional keywords. **MSGVERB** can be set as follows:

MSGVERB=[*keyword*[**:***keyword*[**:** . . .]]]
export MSGVERB

Valid *keywords* are: **label**, **severity**, **text**, **action**, and **tag**. If **MSGVERB** contains a keyword for a component and the component's value is not the component's null value, **fmtmsg( )** includes that component in the message when writing the message to **stderr**. If **MSGVERB** does not include a keyword for a message component, that component is not included in the display of the message. The keywords may appear in any order. If **MSGVERB** is not defined, if its value is the null-string, if its value is not of the correct format, or if it contains keywords other than the valid ones listed above, **fmtmsg( )** selects all components.

The first time **fmtmsg( )** is called, it examines the **MSGVERB** environment variable to see which message components it is to select when generating a message to write to the standard error stream, **stderr**. The values accepted on the initial call are saved for future calls.

MSGVERB affects only which components are selected for display to the standard error stream. All message components are included in console messages.

SEV_LEVEL defines severity levels and associates print strings with them for use by **fmtmsg( ) .** The standard severity levels shown below cannot be modified. Additional severity levels can also be defined, redefined, and removed using **addseverity( )** (see **addseverity**(3C)). If the same severity level is defined by both **SEV_LEVEL** and **addseverity( ) ,** the definition by **addseverity( )** is controlling.

0 (no severity is used)
1 **HALT**
2 **ERROR**
3 **WARNING**
4 **INFO**

SEV_LEVEL can be set as follows:

SEV_LEVEL=[*description*[**:***description*[**:**. . .]]]
export SEV_LEVEL

description is a comma-separated list containing three fields:

*description***=***severity_keyword***,***level***,***printstring*

severity_keyword is a character string that is used as the keyword on the **−s** *severity* option to the **fmtmsg** command. (This field is not used by the **fmtmsg( )** function.)

level is a character string that evaluates to a positive integer (other than 0, 1, 2, 3, or 4, which are reserved for the standard severity levels). If the keyword *severity_keyword* is used, *level* is the severity value passed on to the **fmtmsg( )** function.

printstring is the character string used by **fmtmsg( )** in the standard message format whenever the severity value *level* is used.

If a *description* in the colon list is not a three-field comma list, or, if the second field of a comma list does not evaluate to a positive integer, that *description* in the colon list is ignored.

The first time **fmtmsg( )** is called, it examines the **SEV_LEVEL** environment variable, if defined, to see whether the environment expands the levels of severity beyond the five standard levels and those defined using **addseverity( ) .** The values accepted on the initial call are saved for future calls.

Use in Applications

One or more message components may be systematically omitted from messages generated by an application by using the null value of the argument for that component.

The table below indicates the null values and identifiers for **fmtmsg( )** arguments.

Argument	Type	Null-Value	Identifier
label	**char***	**(char*) NULL**	**MM_NULLLBL**
severity	**int**	**0**	**MM_NULLSEV**
class	**long**	**0L**	**MM_NULLMC**
text	**char***	**(char*) NULL**	**MM_NULLTXT**
action	**char***	**(char*) NULL**	**MM_NULLACT**
tag	**char***	**(char*) NULL**	**MM_NULLTAG**

Another means of systematically omitting a component is by omitting the component keyword(s) when defining the **MSGVERB** environment variable (see the ‘‘Environment Variables’’ section).

RETURN VALUES

The exit codes for **fmtmsg( )** are the following:

MM_OK The function succeeded.

MM_NOTOK The function failed completely.

MM_NOMSG The function was unable to generate a message on the standard error stream, but otherwise succeeded.

MM_NOCON The function was unable to generate a console message, but otherwise succeeded.

EXAMPLES

Example 1:

The following example of **fmtmsg( )**:

> **fmtmsg(MM_PRINT, "UX:cat", MM_ERROR, "invalid syntax", "refer to manual", "UX:cat:001")**

produces a complete message in the standard message format:

```
UX:cat: ERROR: invalid syntax
        TO FIX: refer to manual   UX:cat:001
```

Example 2:

When the environment variable **MSGVERB** is set as follows:

> **MSGVERB=severity:text:action**

and the Example 1 is used, **fmtmsg( )** produces:

```
ERROR: invalid syntax
TO FIX: refer to manual
```

Example 3:

When the environment variable **SEV_LEVEL** is set as follows:

SEV_LEVEL=note,5,NOTE

the following call to **fmtmsg( ) :**

fmtmsg(MM_UTIL | MM_PRINT, "UX:cat", 5, "invalid syntax", "refer to manual", "UX:cat:001")

produces:

```
UX:cat: NOTE: invalid syntax
        TO FIX: refer to manual   UX:cat:001
```

SEE ALSO

fmtmsg(1), **addseverity**(3C), **gettxt**(3C), **printf**(3S)