**NAME**

cmn_err, vcmn_err – display an error message or panic the system

**SYNOPSIS**

**#include <sys/cmn_err.h>**
**#include <sys/ddi.h>**
**#include <sys/sunddi.h>**

**void cmn_err( int** *level*, **char** **format*, **...);**

**void vcmn_err( int** *level*, **char** **format*, **va_list** *ap***);**

**INTERFACE LEVEL**

Architecture independent level 1 (DDI/DKI).

**ARGUMENTS**

**cmn_err( )**

| | |
|---|---|
| *level* | A constant indicating the severity of the error condition. |
| *format* | The message to be displayed. |

**vcmn_err( )**

**vcmn_err( )** takes *level* and *format* as described for **cmn_err( )**, but its third argument is different:

| | |
|---|---|
| *ap* | The variable argument list passed to the function. |

**DESCRIPTION**

**cmn_err( )**

**cmn_err( )** displays a specified message on the console. **cmn_err( )** can also panic the system. When the system panics, it attempts to save recent changes to data, display a "panic message" on the console, attempt to write a core file, and halt system processing. See the **CE_PANIC** *level* below.

*level* is a constant indicating the severity of the error condition. The four severity levels are:

**CE_CONT** Used to continue another message or to display an informative message not associated with an error. Note that multiple **CE_CONT** messages without a newline may or may not appear on the system console or in the system buffer as a single line message. A single line message may be produced by constructing the message with **sprintf**(9F) or **vsprintf**(9F) before calling **cmn_err( )**.

**CE_NOTE** Used to display a message preceded with **NOTICE**. This message is used to report system events that do not necessarily require user action, but may interest the system administrator. For example, a message saying that a sector on a disk needs to be accessed repeatedly before it can be accessed correctly might be noteworthy.

**CE_WARN** Used to display a message preceded with **WARNING**. This message is used to report system events that require immediate attention, such as those where if an action is not taken, the system may panic. For example, when a peripheral device does not initialize correctly, this level should be used.

**CE_PANIC** Used to display a message preceded with **"panic"**, and to panic the system. Drivers should specify this level only under the most severe conditions or when debugging a driver. A valid use of this level is when the system cannot continue to function. If the error is recoverable, or not essential to continued system operation, do not panic the system.

*format* is the message to be displayed. It is a character string which may contain plain characters and conversion specifications. By default, the message is sent both to the system console and to the system buffer.

Each conversion specification in *format* is introduced by the **%** character, after which the following appear in sequence:

An optional decimal digit specifying a minimum field width for numeric conversion. The converted value will be right-justified and padded with leading zeroes if it has fewer characters than the minimum.

An optional **l** (**ll**) specifying that a following **d**, **D**, **o**, **O**, **x**, **X**, or **u** conversion character applies to a **long** (**long long**) integer argument. An **l** (**ll**) before any other conversion character is ignored.

A character indicating the type of conversion to be applied:

**d**,**D**,**o**,**O**,**x**,**X**,**u**
The integer argument is converted to signed decimal (**d**, **D**), unsigned octal (**o**, **O**), unsigned hexadecimal (**x**, **X**), or unsigned decimal (**u**), respectively, and displayed. The letters **abcdef** are used for **x** and **X** conversion.

**c** The character value of the argument is displayed.

**b** The **%b** conversion specification allows bit values to be displayed meaningfully. Each **%b** takes an integer value and a format string from the argument list. The first character of the format string should be the output base encoded as a control character. This base is used to display the integer argument. The remaining groups of characters in the format string consist of a bit number (between 1 and 32, also encoded as a control character) and the next characters (up to the next control character or '\0') give the name of the bit field. The string corresponding to the bit fields set in the integer argument is displayed after the numerical value. See the **EXAMPLES** section.

**s** The argument is taken to be a string (character pointer), and characters from the string are displayed until a null character is encountered. If the character pointer is **NULL**, the string **<null string>** is used in its place.

**%** Copy a %; no argument is converted.

The first character in *format* affects where the message will be written:

**!** the message goes only to the system buffer.

**^** the message goes only to the console.

**?** If *level* is also **CE_CONT**, the message is always sent to the system buffer, but is only written to the console when the system has been booted in verbose mode. See **kernel**(1M). If neither condition is met, the '? ' character has no effect and is simply ignored.

To display the contents of the system buffer, use the **dmesg**(1M) command.

**cmn_err( )** appends a **\n** to each *format*, except when *level* is **CE_CONT**.

**vcmn_err( )**

**vcmn_err( )** is identical to **cmn_err( )** except that its last argument, *ap*, is a pointer to a variable list of arguments. *ap* contains the list of arguments used by the conversion specifications in *format*. *ap* must be initialized by calling **va_start**(9F). **va_end**(9F) is used to clean up and must be called after each traversal of the list. Multiple traversals of the argument list, each bracketed by **va_start**(9F) and **va_end**(9F), are possible.

## RETURN VALUES

None. However, if an unknown *level* is passed to **cmn_err( )**, the following panic error message is displayed:

**panic: unknown level in cmn_err (level=** *level* **, msg=** *format***)**

## CONTEXT

**cmn_err( )** can be called from user or kernel context.

## EXAMPLES

This first example shows how **cmn_err( )** can record tracing and debugging information only in the system buffer (lines 17); display problems with a device only on the system console (line 23); or display problems with the device on both the system console and in the system buffer (line 28).

```
1  struct  reg {
2        uchar_t data;
3        uchar_t csr;
4  };
5
6  struct  xxstate {
7        . . .
8        dev_info_t *dip;
9        struct reg *regp;
10        . . .
11  };
12
13  dev_t dev;
14  struct xxstate *xsp;
15   . . .
16  #ifdef DEBUG    /* in debugging mode, log function call */
17    cmn_err(CE_CONT, "!%s%d: xxopen function called.",
18        ddi_get_name(xsp->dip), getminor(dev));
19  #endif  /* end DEBUG */
20   . . .
21  /* display device power failure on system console */
22    if ((xsp->regp->csr & POWER) == OFF)
```

```
        cmn_err(CE_NOTE, "^%s%d: xxopen: Power is OFF.",
            ddi_get_name(xsp->dip), getminor(dev));
. . .
/* display warning if device has bad VTOC */
  if (xsp->regp->csr & BADVTOC)
        cmn_err(CE_WARN, "%s%d: xxopen: Bad VTOC.",
            ddi_get_name(xsp->dip), getminor(dev));
```

The second example shows how to use the **%b** conversion specification. Because of the leading '? ' character in the format string, this message will always be logged, but it will only be displayed when the kernel is booted in verbose mode.

```
cmn_err(CE_CONT, "?reg=0x%b\n", regval, "\020\3Intr\2Err\1Enable");
```

When *regval* is set to (decimal) **13**, the following message would be displayed:

```
reg=0xd<Intr,,Enable>
```

The third example is an error reporting routine which accepts a variable number of arguments and displays a single line error message both in the system buffer and on the system console. Note the use of **vsprintf( )** to construct the error message before calling **cmn_err( )**.

```
#include <sys/varargs.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>

#define MAX_MSG 256

void
xxerror(dev_info_t *dip, int level, const char *fmt, . . . )
{
     va_list      ap;
     int          instance;
     char          buf[MAX_MSG],
                  *name;

     instance = ddi_get_instance(dip);
     name = ddi_get_name(dip);

     /* format buf using fmt and arguments contained in ap */
     va_start(ap, fmt);
     vsprintf(buf, fmt, ap);
     va_end(ap);

     /* pass formatted string to cmn_err(9F) */
     cmn_err(level, "%s%d: %s", name, instance, buf);
}
```

**SEE ALSO**

**dmesg**(1M), **kernel**(1M), **printf**(3S), **sprintf**(9F), **va_arg**(9F), **vsprintf**(9F)

*Writing Device Drivers*

**WARNINGS**

**cmn_err( )** with the **CE_CONT** argument can be used by driver developers as a driver code debugging tool. However, using **cmn_err( )** in this capacity can change system timing characteristics.

**NOTES**

At times, a driver may encounter error conditions requiring the attention of a primary or secondary system console monitor. These conditions may mean halting multiuser processing; however, this must be done with caution. Except during the debugging stage, a driver should never stop the system.

See the "Debugging" chapter in *Writing Device Drivers*.

**BUGS**

**cmn_err( )** does not provide all of the functionality provided by **printf**(3S).