**NAME**

vprintf, vfprintf, vsprintf – print formatted output of a variable argument list

**SYNOPSIS**

**#include <stdio.h>**
**#include <stdarg.h>**

**int vprintf(const char** **format*, **va_list** *ap***);**

**int vfprintf(FILE** **stream*, **const char** **format*, **va_list** *ap***);**

**int vsprintf(char** **s*, **const char** **format*, **va_list** *ap***);**

**MT-LEVEL**

See the **NOTES** section of this page.

**DESCRIPTION**

**vprintf( )**, **vfprintf( )**, and **vsprintf( )** are the same as **printf( )**, **fprintf( )**, and **sprintf( )** respectively, except that instead of being called with a variable number of arguments, they are called with an argument list as defined by the **<stdarg.h>** header.

The **<stdarg.h>** header defines the type **va_list** and a set of macros for advancing through a list of arguments whose number and types may vary. The argument *ap* to the vprint family of routines is of type **va_list**. This argument is used with the **<stdarg.h>** header file macros **va_start( )**, **va_arg( )**, and **va_end( )** (see **stdarg**(5)). The **EXAMPLES** section below shows the use of **va_start( )** and **va_end( )** with **vprintf( )**.

The macro **va_alist** is used as the parameter list in a function definition, as in the function called **error( )** in the example below. The macro **va_start(***ap, parmN***)**, where *ap* is of type **va_list**, and *parmN* is the rightmost parameter (just before . . .), must be called before any attempt to traverse and access unnamed arguments is made. The **va_end(***ap***)** macro must be invoked when all desired arguments have been accessed. (The argument list in *ap* can be traversed again if **va_start( )** is called again after **va_end( )**.) In the example below, the **error( )** arguments, *arg1, arg2, . . .*, are given to **vfprintf( )** in the argument *ap*.

**RETURN VALUES**

**vprintf( )** and **vfprintf( )** return the number of characters transmitted, or return −**1** if an error was encountered.

**EXAMPLES**

The following demonstrates how **vfprintf( )** could be used to write an **error** routine:

```
#include <stdio.h>
#include <stdarg.h>
. . .
/*
 *  error should be called like
 *       error(function_name, format, arg1, . . .);
 */
void error(char *function_name, char *format, . . .)
{
   va_list ap;
   va_start(ap, );
   /* print out name of function causing error */
   (void) fprintf(stderr, "ERR in %s: ", function_name);
```

```
        /* print out remainder of message */
        (void) vfprintf(stderr, format, ap);
        va_end(ap);
        (void) abort;
    }
```

**SEE ALSO** **printf**(3S), **stdarg**(5)

**NOTES** **vprintf( )**, **vfprintf( )**, and **vsprintf( )** are MT-Safe in multi-thread applications.