its output and always writes to stderr instead of stdout. We'll have errorf call vfprintf to do most of the actual output. Here's what errorf might look like:

```
int errorf(const char *format, ...)
{
  static int num_errors = 0;
  int n;
  va_list ap;

  num_errors++;
  fprintf(stderr, "** Error %d: ", num_errors);
  va_start(ap, format);
  n = vfprintf(stderr, format, ap);
  va_end(ap);
  fprintf(stderr, "\n");
  return n;
}
```

The wrapper function—errorf, in our example—is responsible for calling va_start prior to calling the v...printf function and for calling va_end after the v...printf function returns. The wrapper function is allowed to call va_arg one or more times before calling the v...printf function.

vsnprintf The vsnprintf function was added to the C99 version of <stdio.h>. It corresponds to snprintf (discussed in Section 22.8), which is also a C99 function.

C99 The v...scanf Functions

```
int vfscanf(FILE * restrict stream,
            const char * restrict format,
            va_list arg);                    from <stdio.h>
int vscanf(const char * restrict format,
           va_list arg);                     from <stdio.h>
int vsscanf(const char * restrict s,
            const char * restrict format,
            va_list arg);                    from <stdio.h>
```

vfscanf *vscanf* *vsscanf* C99 adds a set of "v...scanf functions" to the <stdio.h> header. vfscanf, vscanf, and vsscanf are equivalent to fscanf, scanf, and sscanf, respectively, except that they have a va_list parameter through which a variable argument list can be passed. Like the v...printf functions, each v...scanf function is designed to be called by a wrapper function that accepts a variable number of arguments, which it then passes to the v...scanf function. The wrapper function is responsible for calling va_start prior to calling the v...scanf function and for calling va_end after the v...scanf function returns.