The #ifndef directive is similar to #ifdef, but tests whether an identifier is *not* defined as a macro:

#ifndef directive

#ifndef *identifier*

Writing

#ifndef *identifier*

is the same as writing

#if !defined(*identifier*)

The #elif and #else Directives

#if, #ifdef, and #ifndef blocks can be nested just like ordinary if statements. When nesting occurs, it's a good idea to use an increasing amount of indentation as the level of nesting grows. Some programmers put a comment on each closing #endif to indicate what condition the matching #if tests:

```
#if DEBUG
...
#endif /* DEBUG */
```

This technique makes it easier for the reader to find the beginning of the #if block.

For additional convenience, the preprocessor supports the #elif and #else directives:

#elif directive

#elif *constant-expression*

#else directive

#else

#elif and #else can be used in conjunction with #if, #ifdef, or #ifndef to test a series of conditions:

```
#if expr1
Lines to be included if expr1 is nonzero
#elif expr2
Lines to be included if expr1 is zero but expr2 is nonzero
#else
Lines to be included otherwise
#endif
```

Although the #if directive is shown above, an #ifdef or #ifndef directive can be used instead. Any number of #elif directives—but at most one #else—may appear between #if and #endif.