```
#if INT_MAX < 100000
#error int type is too small
#endif
```

Attempting to compile the program on a machine whose integers are stored in 16 bits will produce a message such as

```
Error directive: int type is too small
```

The `#error` directive is often found in the `#else` part of an `#if`-`#elif`-`#else` series:

```
#if defined(WIN32)
...
#elif defined(MAC_OS)
...
#elif defined(LINUX)
...
#else
#error No operating system specified
#endif
```

The `#line` Directive

The `#line` directive is used to alter the way program lines are numbered. (Lines are usually numbered 1, 2, 3, as you'd expect.) We can also use this directive to make the compiler think that it's reading the program from a file with a different name.

The `#line` directive has two forms. In one form, we specify a line number:

`#line` directive (form 1)

```
#line n
```

n must be a sequence of digits representing an integer between 1 and 32767 (2147483647 in C99). This directive causes subsequent lines in the program to be numbered *n*, *n* + 1, *n* + 2, and so forth.

C99

In the second form of the `#line` directive, both a line number and a file name are specified:

`#line` directive (form 2)

```
#line n "file"
```

The lines that follow this directive are assumed to come from *file*, with line numbers starting at *n*. The values of *n* and/or the *file* string can be specified using macros.

One effect of the `#line` directive is to change the value of the `__LINE__` macro (and possibly the `__FILE__` macro). More importantly, most compilers will use the information from the `#line` directive when generating error messages.