sage that corresponds to the EDOM error. An ERANGE error usually produces a different message, such as `Numerical result out of range`.

strerror The `strerror` function belongs to `<string.h>`. When passed an error code, `strerror` returns a pointer to a string describing the error. For example, the call

```
puts(strerror(EDOM));
```

might print

```
Numerical argument out of domain
```

The argument to `strerror` is usually one of the values of `errno`, but `strerror` will return a string for any integer passed to it.

`strerror` is closely related to the `perror` function. The error message that `perror` displays is the same message that `strerror` would return if passed `errno` as its argument.

## 24.3 The `<signal.h>` Header: Signal Handling

The `<signal.h>` header provides facilities for handling exceptional conditions, known as ***signals***. Signals fall into two categories: run-time errors (such as division by zero) and events caused outside the program. Many operating systems, for example, allow users to interrupt or kill running programs; these events are treated as signals in C. When an error or external event occurs, we say that a signal has been ***raised***. Many signals are asynchronous: they can happen at any time during program execution, not just at certain points that are known to the programmer. Since signals may occur at unexpected times, they have to be dealt with in a unique way.

This section covers signals as they're described in the C standard. Signals play a more prominent role in UNIX than you might expect from their limited coverage here. For information about UNIX signals, consult one of the UNIX programming books listed in the bibliography.

### Signal Macros

Q&A `<signal.h>` defines a number of macros that represent signals; Table 24.1 lists these macros and their meanings. The value of each macro is a positive integer constant. C implementations are allowed to provide other signal macros, as long as their names begin with `SIG` followed by an upper-case letter. (UNIX implementations, in particular, provide a large number of additional signal macros.)

The C standard doesn't require that the signals in Table 24.1 be raised automatically, since not all of them may be meaningful for a particular computer and operating system. Most implementations support at least some of these signals.