Errors from System Calls

System calls can fail in a variety of ways. For example, you may try to read a file that does not exist or you do not have permissions for. You might also try to connect to a remote computer across the network at an address that is invalid or unreachable (the network or remote computer is down). Whenever these system calls fail in C, they (or technically their wrapper in the C library) set a global variable called **errno** (which stands for "error number").

We have not worked with global variables—and their use is typically discouraged—but they are simply variables whose scope is the entire program. They are declared outside of any function, and have a "box" which is not part of any frame. You can read (or write) them just like any other variable.

In the particular case of **errno**, it is set by a failing call, and read if your program wants more information about *why* the call failed. If you want to check if a specific error occurred, you can compare it against the various constants which are defined in **errno.h**.

You may also wish to print out a message describing the error for the user. Just printing the numeric value of **errno**is not usually useful (Do you know what error 2 means?). Fortunately, the C library has a function **perror**, which prints a descriptive error message based on the current value of **errno**. The **perror**function takes one argument, a string that it prints before its descriptive message.

Note that since **errno**is global (there is *one* for the entire program), you must take care not to call anything that might change it before you test it or call **perror**. For example,

```
6

3

4

5

1

2

}

if (x != 0) {

  printf("someSystemCall() failed!\n");  //may change errno

  perror("The error was: ");
```

```
//BROKEN CODE

int x = someSystemCall();
```

Here, **printf**might change **errno** (it makes system calls), so we may not have a correct description of the error from **perror**. The possibility of unexpected changes from other parts of the code is one of the hazards of global variables, and part of the reason to avoid them in general.