To remove a file named *a confusing name*, the first **rm** command, which follows, doesn't work; the second one does:

```
$ ls -l
total 2
-rw-r--r--  1  jpeek users     0 Oct 23 11:23 a confusing name
-rw-r--r--  1  jpeek users  1674  Oct 23 11:23 ch01
$ rm a confusing name
rm: a: no such file or directory
rm: confusing: no such file or directory
rm: name: no such file or directory
$ rm "a confusing name"
$
```

Unlike some operating systems, Unix doesn't require a dot (`.`) in a filename; in fact, you can use as many as you want. For instance, the filenames *pizza* and *this.is.a.mess* are both legal.

Some Unix systems limit filenames to 14 characters. Most newer systems allow much longer filenames.

A filename must be unique inside its directory, but other directories may have files with the same names. For example, you may have the files called *chap1* and *chap2* in the directory */users/carol/work* and also have files with the same names in */users/carol/play*.

File and Directory Wildcards

When you have a number of files named in series (for example, *chap1* to *chap12*) or filenames with common characters (such as *aegis*, *aeon*, and *aerie*), you can use *wildcards* to specify many files at once. These special characters are `*` (asterisk), `?` (question mark), and `[ ]` (square brackets). When used in a file or directory name given as an argument on a command line, the following is true:

`*` An asterisk stands for any number of characters in a filename. For example, *ae** would match *aegis*, *aerie*, *aeon*, etc. if those files were in the same directory. You can use this to save typing for a single filename (for example, *al** for *alphabet.txt*) or to choose many files at once (as in *ae**). A `*` by itself matches all file and subdirectory names in a directory.

`?` A question mark stands for any single character (so *h?p* matches *hop* and *hip*, but not *help*).