`[]` Square brackets can surround a choice of single characters (i.e., one digit or one letter) you'd like to match. For example, [*Cc*]*hapter* would match either *Chapter* or *chapter*, but [*ch*]*apter* would match either *capter* or *hapter*. Use a hyphen (–) to separate a range of consecutive characters. For example, *chap[1–3]* would match *chap1*, *chap2*, or *chap3*.

The following examples show the use of wildcards. The first command lists all the entries in a directory, and the rest use wildcards to list just some of the entries. The last one is a little tricky; it matches files whose names contain two (or more) *a*'s.

```
$ ls
chap10       chap2        chap5     cold
chap1a.old   chap3.old    chap6     haha
chap1b       chap4        chap7     oldjunk
$ ls chap?
chap2    chap5    chap7
chap4    chap6
$ ls chap[5-7]
chap5    chap6    chap7
$ ls chap[5-9]
chap5    chap6    chap7
$ ls chap??
chap10   chap1b
$ ls *old
chap1a.old   chap3.old    cold
$ ls *a*a*
chap1a.old   haha
```

Wildcards are useful for more than listing files. Most Unix programs accept more than one filename, and you can use wildcards to name multiple files on the command line. For example, the **less** program displays a file on the screen. Let's say you want to display files *chap3.old* and *chap1a.old*. Instead of specifying these files individually, you could enter the command as:

```
$ less *.old
```

This is equivalent to "**less chap1a.old chap3.old**".

Wildcards match directory names, too. You can use them anywhere in a pathname—absolute or relative—though you still need to separate directory levels with slashes (/). For example, let's say you have subdirectories named *Jan*, *Feb*, *Mar*, and so on. Each has a file named *summary*. You could read all the summary files by typing "**less */summary**". That's almost