### 24.1.10 How do I read and write the serial port?

This depends on which operating system your program is running on. In the case of Unix, the serial ports will be accessible through files in /dev; on other systems, device names will doubtless differ. Several problem areas common to all device interaction are the following:

**lockfiles**

Your system may use lockfiles to control multiple access. Make sure you follow the correct protocol. Unpredictable behavior can result from multiple processes reading from one device.

**open mode**

If you expect to use both read and write operations on the device, you'll have to open it for update (see open in *perlfunc* for details). You may wish to open it without running the risk of blocking by using sysopen() and `O_RDWR|O_NDELAY|O_NOCTTY` from the Fcntl module (part of the standard perl distribution). See sysopen in *perlfunc* for more on this approach.

**end of line**

Some devices will be expecting a "\r" at the end of each line rather than a "\n". In some ports of perl, "\r" and "\n" are different from their usual (Unix) ASCII values of "\012" and "\015". You may have to give the numeric values you want directly, using octal ("\015"), hex ("0x0D"), or as a control-character specification ("\cM").

```
print DEV "atv1\012";       # wrong, for some devices
print DEV "atv1\015";       # right, for some devices
```

Even though with normal text files a "\n" will do the trick, there is still no unified scheme for terminating a line that is portable between Unix, DOS/Win, and Macintosh, except to terminate *ALL* line ends with "\015\012", and strip what you don't need from the output. This applies especially to socket I/O and autoflushing, discussed next.

**flushing output**

If you expect characters to get to your device when you print() them, you'll want to autoflush that filehandle. You can use select() and the $| variable to control autoflushing (see $| in *perlvar* and select in *perlfunc*, or *perlfaq5*, "How do I flush/unbuffer an output filehandle? Why must I do this?"):

```
$oldh = select(DEV);
$| = 1;
select($oldh);
```

You'll also see code that does this without a temporary variable, as in

```
select((select(DEV), $| = 1)[0]);
```

Or if you don't mind pulling in a few thousand lines of code just because you're afraid of a little $| variable:

```
use IO::Handle;
DEV->autoflush(1);
```

As mentioned in the previous item, this still doesn't work when using socket I/O between Unix and Macintosh. You'll need to hard code your line terminators, in that case.

**non-blocking input**

If you are doing a blocking read() or sysread(), you'll have to arrange for an alarm handler to provide a timeout (see alarm in *perlfunc*). If you have a non-blocking open, you'll likely have a non-blocking read, which means you may have to use a 4-arg select() to determine whether I/O is ready on that device (see select in *perlfunc*.