Proto

> This is which protocol to use. In this case, the socket handle returned will be connected to a TCP socket, because we want a stream-oriented connection, that is, one that acts pretty much like a plain old file. Not all sockets are this of this type. For example, the UDP protocol can be used to make a datagram socket, used for message-passing.

PeerAddr

> This is the name or Internet address of the remote host the server is running on. We could have specified a longer name like `"www.perl.com"`, or an address like `"204.148.40.9"`. For demonstration purposes, we've used the special hostname `"localhost"`, which should always mean the current machine you're running on. The corresponding Internet address for localhost is `"127.1"`, if you'd rather use that.

PeerPort

> This is the service name or port number we'd like to connect to. We could have gotten away with using just `"daytime"` on systems with a well-configured system services file,[FOOTNOTE: The system services file is in */etc/services* under Unix] but just in case, we've specified the port number (13) in parentheses. Using just the number would also have worked, but constant numbers make careful programmers nervous.

Notice how the return value from the `new` constructor is used as a filehandle in the `while` loop? That's what's called an indirect filehandle, a scalar variable containing a filehandle. You can use it the same way you would a normal filehandle. For example, you can read one line from it this way:

```
$line = <$handle>;
```

all remaining lines from is this way:

```
@lines = <$handle>;
```

and send a line of data to it this way:

```
print $handle "some data\n";
```

A Webget Client

Here's a simple client that takes a remote host to fetch a document from, and then a list of documents to get from that host. This is a more interesting client than the previous one because it first sends something to the server before fetching the server's response.

```
#!/usr/bin/perl -w
use IO::Socket;
unless (@ARGV > 1) { die "usage: $0 host document ..." }
$host = shift(@ARGV);
$EOL = "\015\012";
$BLANK = $EOL x 2;
foreach $document ( @ARGV ) {
    $remote = IO::Socket::INET->new( Proto     => "tcp",
                                     PeerAddr  => $host,
                                     PeerPort  => "http(80)",
                                   );
    unless ($remote) { die "cannot connect to http daemon on $host" }
    $remote->autoflush(1);
    print $remote "GET $document HTTP/1.0" . $BLANK;
    while ( <$remote> ) { print }
    close $remote;
}
```