

Sockets 1

This first Perl/Socket handout just introduce the most basic Perl/Socket material.

Sockets are a pretty common interface that make TCP streams look like file streams. This code shows the Perl versions, but it looks pretty similar in other languages. A Perl program needs the statement "use Socket;" up near its top to import these functions.

1. Hostname Lookup

`$ipaddr = inet_aton($hostname)` -- tries to look up the IP address of a hostname such as "www.yahoo.com". This will probably do a quick transaction with the local DNS server to get the IP address. Fails if the hostname cannot be looked up either because the hostname is wrong, or the local DNS system is down. Returns false on error and sets \$!.

2. Create Socket Address

`$sockaddr = sockaddr_in($port, $ipaddr)`; -- given a port number and an ipaddr, combine the two into a socket address that can be passed to `connect()`. We'll assume this step doesn't fail.

3. Allocate Socket

`socket($sock, PF_INET, SOCK_STREAM, 0)` -- given the name to use for the socket handle in `$sock`, such as 'SOCK' (the quotes are optional actually), allocate a socket. This just allocates the socket resources; it does not try to connect the socket to anything. The constants shown are to create a stream (virtual circuit) connection using TCP. Other constants can be used to create other sorts of sockets. Fails if there are not enough resources to create another socket on the local machine. Returns false on error and sets \$!.

4. Connect

`connect($sock, $sockaddr)` -- given a socket (step 3) and a socket address (step 2), try to connect to the given port on that machine. On success, the given socket can now be used for reading and writing to the remote machine. Reading will block if there is no data available. Fails if the connection cannot be made to the remote machine because it is unreachable or it is not listening on that port number. Returns false on error and sets \$!.

5 Read

The simplest way to read text data is just with the standard `<SOCK>` file reading operator. Returns undef on error or the EOF.

```
connect(SOCK, $sockaddr);  
$line = <SOCK>;    ## read a line of text
```

6 Write

The simplest way to write text is data is with print...

```
print SOCK "Hello there, how are you?\012";
```

7 Close

close(SOCK) -- closes the reading and writing capabilities of this end of the socket. The close on the writing end here will show up as an EOF on the other end.