Introducing The Perl Socket API

```
#! /usr/bin/perl -w

use strict;
use Socket;
```

Socket Utility Subroutines - 1 of 4

```
my $ip = '127.0.0.1';
my $binary = inet_aton( $ip );
my $dotted_decimal = inet_ntoa( $binary );
```

Socket Utility Subroutines - 2 of 4

```
my $ip = '127.0.0.1';
my $port = 80;
my $binary = inet_aton( $ip );
my $socket_addr = sockaddr_in( $port, $binary );
my ( $socket_port, $socket_binary ) = sockaddr_in( $socket_addr );
my $socket_dotted_decimal = inet_ntoa( $socket_binary );
```

Socket Utility Subroutines - 3 of 4

```
my $port = 80;

    ...

my $port = getservbyname( 'http', 'tcp' );

    ...

my $trans_service_id = getprotobyname( 'udp' );
```

Socket Utility Subroutines - 4 of 4

```
my $packed_binary_addr = gethostbyname( 'tyndall.itcarlow.ie' );
my $ip_addr = inet_ntoa( $packed_binary_addr );
print "tyndall's IP address is: $ip_addr\n";
my $ip_name = gethostbyaddr( $packed_binary_addr, AF_INET );
print "$ip_addr has the following name: $ip_name\n";

    ...

tyndall's IP address is: 149.153.1.5
149.153.1.5 has the following name: ns.itcarlow.ie
```

The First UDP Server - 1 of 2

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_UDP_PORT => 4001;
use constant MAX_RECV_LEN    => 1500;
use constant LOCAL_INETNAME  => 'localhost';

my $trans_serv = getprotobyname( 'udp' );

my $local_host = gethostbyname( LOCAL_INETNAME );
my $local_port = SIMPLE_UDP_PORT;
my $local_addr = sockaddr_in( $local_port, $local_host );
```

The First UDP Server - 2 of 2

```
socket( UDP_SOCK, PF_INET, SOCK_DGRAM, $trans_serv );

bind( UDP_SOCK, $local_addr );

my $data;

while( 1 )
{
    my $from_who = recv( UDP_SOCK, $data, MAX_RECV_LEN, 0 );

    if ( $from_who )
    {
        my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

        warn 'Received from ', inet_ntoa( $the_ip ), ": $data\n";
    }
    else
    {
        warn "Problem with recv: $!\n";
    }
}
```

The First UDP Client

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_UDP_PORT => 4001;
use constant REMOTE_HOST     => 'localhost';

my $trans_serv  = getprotobyname( 'udp' );
my $remote_host = gethostbyname( REMOTE_HOST );
my $remote_port = SIMPLE_UDP_PORT;
my $destination = sockaddr_in( $remote_port, $remote_host );

socket( UDP_SOCK, PF_INET, SOCK_DGRAM, $trans_serv );

my $data = "This is a simple UDP message";

send( UDP_SOCK, $data, 0, $destination );

close UDP_SOCK;
```

Genericity and Robustness - On The Client

```
my $local_port = shift || SIMPLE_UDP_PORT;

    ...

my $local_addr = sockaddr_in( $local_port, INADDR_ANY );

    ...

my $remote = shift || REMOTE_HOST;
my $remote_port = shift || SIMPLE_UDP_PORT;

    ...

my $remote_host = gethostbyname( $remote )
```

Genericity and Robustness On The Server - 1 of 3

```
socket( UDP_SOCK, PF_INET, SOCK_DGRAM, $trans_serv )
    or die "udp_s2: socket creation failed: $!\n";

bind( UDP_SOCK, $local_addr )
    or die "udp_s2: bind to address failed: $!\n";
```

Genericity and Robustness On The Server - 2 of 3

```
my $remote_host = gethostbyname( $remote )
    or die "udp_c2: name lookup failed: $remote\n";

socket( UDP_SOCK, PF_INET, SOCK_DGRAM, $trans_serv )
    or die "udp_c2: socket creation failed: $!\n";

send( UDP_SOCK, $data, 0, $destination )
    or warn "udp_c2: send to socket failed.\n";

close UDP_SOCK
    or die "udp_c2: close socket failed: $!\n";
```

Genericity and Robustness
On The Server - 3 of 3

```
if ( $from_who )
{
    my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

    my $remote_name = gethostbyaddr( $the_ip, AF_INET );

    warn "Received from $remote_name: $data\n";
}
```

Genericity and Robustness - Example Output

```
Received from glasnost.itcarlow.ie: This is a simple UDP message
Received from localhost: This is a simple UDP message
Received from localhost: This is a simple UDP message
Received from glasnost.itcarlow.ie: This is a simple UDP message
Received from localhost: This is a simple UDP message
   ...
```

UDP Is Unreliable

```
my $msg_count = 1;

while ( $msg_count < 6 )
{
    my $data = "This is a simple UDP message, number $msg_count";

    send( UDP_SOCK, $data, 0, $destination )
        or warn "udp_c3: send to socket failed: $msg_count\n";

    $msg_count++;
}

close UDP_SOCK
    or die "udp_c3: close socket failed: $!\n";
```

UDP Is Unreliable
Client and Server Available

```
Received from localhost: This is a simple UDP message, number 1
Received from localhost: This is a simple UDP message, number 2
Received from localhost: This is a simple UDP message, number 3
Received from localhost: This is a simple UDP message, number 4
Received from localhost: This is a simple UDP message, number 5
```

UDP Is Unreliable
No Server Available

```
udp_c3: send to socket failed: 2
udp_c3: send to socket failed: 4
```

No Flow Control in UDP - 1 of 2

```
# Slow Server.

warn "Received from $remote_name: $data\n";

sleep(3);
```

No Flow Control in UDP - 2 of 2

```
# Three Times Faster Client.

my $big_chunk = 'x' x 65000;

while ( $msg_count < 11 )
{
    my $data = $msg_count . ' -> ' . $big_chunk;

    send( UDP_SOCK, $data, 0, $destination )
        or warn "udp_c4: send to socket failed: $msg_count\n";

    sleep(1);

    $msg_count++;
}
```

No Flow Control - Example Output

```
Received from localhost: 1448 -> 1 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 1448 -> 2 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 1448 -> 4 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 1448 -> 7 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 1448 -> 10 -> xxxxxxxxxxxxxxxxxxxxxxxxxx

    ...

use constant MAX_RECV_LEN    => 65536;
```

Sending and Receiving with UDP

```
# On the Server.
send( UDP_SOCK, $data, 0, $from_who )
    or warn "udp_s5: send to socket failed.\n";

    ...
# On the Client.
my $from_who = recv( UDP_SOCK, $data, MAX_RECV_LEN, 0 );

my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

my $remote_name = gethostbyaddr( $the_ip, AF_INET );

warn "Received from $remote_name: ", length( $data ),
     ' -> ', substr( $data, 0, 39 ), "\n";
```

Sending and Receiving with UDP - On `localhost`

```
Received from localhost: 65005 -> 1 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 2 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 3 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 4 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 5 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 6 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 7 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 8 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65005 -> 9 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
Received from localhost: 65006 -> 10 -> xxxxxxxxxxxxxxxxxxxxxxxxxx
```

Sending and Receiving with UDP Over A Network - 1 of 2

```
Server starting up on port: 4001.
Received from linux303.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxxx
Sending back to client ...
```

Sending and Receiving with UDP Over A Network - 2 of 2

```
Sending 1 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxxxxx
Sending 2 to server ...


# Client is waiting for Server.
# Server is waiting for Client.
# Results in Classic Deadlock.
```

Dealing with Deadlock
Specifying a Time-out on the Client - 1 of 2

```
$SIG{ALRM} = sub { die "recv timeout\n"; };

alarm( 5 );

eval {
    my $from_who = recv( UDP_SOCK, $data, MAX_RECV_LEN, 0 );

    if ( $from_who )
    {
        my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

        my $remote_name = gethostbyaddr( $the_ip, AF_INET );

        warn "Received from $remote_name: ", length( $data ),
            ' -> ', substr( $data, 0, 39 ), "\n";
    }
    else
    {
        warn "Problem with recv: $!\n";
    }
    alarm( 0 );
};
```

Dealing with Deadlock
Specifying a Time-out on the Server - 2 of 2

```
if ($@)
{
    die "udp_c6: $@\n" unless $@ =~ /recv timeout/;

    warn "udp_c6: recv timed out, canceling ...\n";
}
```

Specifying a Time-out - Example Output, Server

```
Server starting up on port: 4001.
Received from linux303.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 3 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 4 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 5 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 6 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 7 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 8 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 9 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65006 -> 10 -> xxxxxxxxxxxx
Sending back to client ...
```

Specifying a Time-out - Example Output, Client

```
Sending 1 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxxx
Sending 2 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxx
Sending 3 to server ...
udp_c6: recv timed out, canceling ...
Sending 4 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 4 -> xxxxxxxxxxxxx
Sending 5 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 5 -> xxxxxxxxxxxxx
Sending 6 to server ...
udp_c6: recv timed out, canceling ...
Sending 7 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 7 -> xxxxxxxxxxxxx
Sending 8 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 8 -> xxxxxxxxxxxxx
Sending 9 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 9 -> xxxxxxxxxxxxx
Sending 10 to server ...
Received from pblinux.itcarlow.ie: 65006 -> 10 -> xxxxxxxxxxxx
```

Dealing with Deadlock
Checking for Data on the Client - 1 of 2

```
my $read_bits = '';

vec( $read_bits, fileno( UDP_SOCK ), 1 ) = 1;

my $recv_ok = select( $read_bits, undef, undef, 0 );

if ( $recv_ok )
{
    my $from_who = recv( UDP_SOCK, $data, MAX_RECV_LEN, 0 );

    if ( $from_who )
    {
        my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

        my $remote_name = gethostbyaddr( $the_ip, AF_INET );

        warn "Received from $remote_name: ", length( $data ),
            ' -> ', substr( $data, 0, 39 ), "\n";
    }
```

Dealing with Deadlock
Checking for Data on the Client - 2 of 2

```
    else
    {
        warn "Problem with recv: $!\n";
    }
}
else
{
    warn "udp_c7: recv skipped, no data is waiting.\n";
}
```

Checking for Data - Example Output, Server

```
Server starting up on port: 4001.
Received from linux303.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 5 -> xxxxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 8 -> xxxxxxxxxxxxxxx
Sending back to client ...
```

Checking for Data - Example Output, Client

```
Sending 1 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 2 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 3 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 4 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 5 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 6 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 7 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxxx
Sending 8 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 9 to server ...
udp_c7: recv skipped, no data is waiting.
Sending 10 to server ...
udp_c7: recv skipped, no data is waiting.
```

Dealing with Deadlock
Spawning a Sub-process - 1 of 7

```
my $child_pid = fork;

if ( $child_pid )
{
    continue_as_parent( $child_pid );
}
elsif ( defined( $child_pid ) )
{
    continue_as_child;
    exit;
}
else
{
    warn "fork failed: $!\n";
}
```

Dealing with Deadlock
Spawning a Sub-process - 2 of 7

```
use POSIX ":sys_wait_h";

sub zombie_reaper {
    while ( waitpid( -1, WNOHANG ) > 0 )
    { }
    $SIG{CHLD} = \&zombie_reaper;
}

$SIG{CHLD} = \&zombie_reaper;
```

Dealing with Deadlock
Spawning a Sub-process - 3 of 7

```
if ( kill 0 => $child_pid )
{
    warn "udp_c8: some child is already executing.\n";
}
else
{
    warn "udp_c8: the child is dead, zombied or " .
         "now belongs to some other user.\n";
}
```

Dealing with Deadlock
Spawning a Sub-process - 4 of 7

```
no strict 'vars';

if ( defined( $child_pid ) )
{
    if ( kill 0 => $child_pid )
    {
        next;
    }
}

$child_pid = fork;

if ( $child_pid )
{
    next;
}
```

Dealing with Deadlock
Spawning a Sub-process - 5 of 7

```
elsif ( defined( $child_pid ) )
{
    my $from_who = recv( UDP_SOCK, $data, MAX_RECV_LEN, 0 );

    if ( $from_who )
    {
        my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

        my $remote_name = gethostbyaddr( $the_ip, AF_INET );

        warn "Received from $remote_name: ", length( $data ),
            ' -> ', substr( $data, 0, 39 ), "\n";
    }
```

Dealing with Deadlock
Spawning a Sub-process - 6 of 7

```
    else
    {
        warn "Problem with recv: $!\n";
    }

    exit;
}
else
{
    warn "udp_c8: fork failed: $!\n";
}

use strict 'vars';
```

Dealing with Deadlock
Spawning a Sub-process - 7 of 7

```
sleep( 5 );

if ( kill 0 => $child_pid )
{
    kill 9 => $child_pid;
}
```

Spawning a Sub-process - Example Output, Server

```
Server starting up on port: 4001.
Received from linux303.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 5 -> xxxxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 65005 -> 8 -> xxxxxxxxxxxxxxx
Sending back to client ...
Problem with recv: Connection refused.
```

Spawning a Sub-process - Example Output, Client

```
Sending 1 to server ...
Sending 2 to server ...
Sending 3 to server ...
Sending 4 to server ...
Sending 5 to server ...
Sending 6 to server ...
Sending 7 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxx
Sending 8 to server ...
Sending 9 to server ...
Sending 10 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 5 -> xxxxxxxxxxxxx
```

The First TCP Server - 1 of 4

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_TCP_PORT => 4001;
use constant MAX_RECV_LEN    => 65536;

my $local_port = shift || SIMPLE_TCP_PORT;

my $trans_serv = getprotobyname( 'tcp' );

my $local_addr = sockaddr_in( $local_port, INADDR_ANY );
```

The First TCP Server - 2 of 4

```
socket( TCP_SOCK, PF_INET, SOCK_STREAM, $trans_serv )
    or die "tcp_s1: socket creation failed: $!\n";

setsockopt( TCP_SOCK, SOL_SOCKET, SO_REUSEADDR, 1 )
    or warn "tcp_s1: could not set socket option: $!\n";

bind( TCP_SOCK, $local_addr )
    or die "tcp_s1: bind to address failed: $!\n";

listen( TCP_SOCK, SOMAXCONN )
    or die "tcp_s1: listen couldn't: $!\n";

warn "Server starting up on port: $local_port.\n";

my $from_who;
```

The First TCP Server - 3 of 4

```
while ( $from_who = accept( CLIENT_SOCK, TCP_SOCK ) )
{
    my $data;

    $from_who = recv( CLIENT_SOCK, $data, MAX_RECV_LEN, 0 );

    if ( $from_who )
    {
        my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

        my $remote_name = gethostbyaddr( $the_ip, AF_INET );

        warn "Received from $remote_name: ", length( $data ),
            ' -> ', substr( $data, 0, 39 ), "\n";
    }
    else
    {
        warn "tcp_s1: problem with recv: $!\n";
        next;
    }
```

The First TCP Server - 4 of 4

```
    sleep( 3 );

    warn "Sending back to client ... \n";

    send( CLIENT_SOCK, $data, 0 )
        or warn "tcp_s1: problem with send: $!\n";
}
continue {
    close CLIENT_SOCK
        or warn "tcp_s1: close failed: $!\n";
}

close TCP_SOCK;
```

The First TCP Client - 1 of 3

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_TCP_PORT => 4001;
use constant REMOTE_HOST     => 'localhost';
use constant MAX_RECV_LEN    => 65536;

my $remote = shift || REMOTE_HOST;
my $remote_port = shift || SIMPLE_TCP_PORT;

my $trans_serv  = getprotobyname( 'tcp' );

my $remote_host = gethostbyname( $remote )
    or die "tcp_c1: name lookup failed: $remote\n";

my $destination = sockaddr_in( $remote_port, $remote_host );
```

The First TCP Client - 2 of 3

```
my $msg_count = 1;

my $big_chunk = 'x' x 65000;

while ( $msg_count < 11 )
{
    socket( TCP_SOCK, PF_INET, SOCK_STREAM, $trans_serv )
        or die "tcp_c1: socket creation failed: $!\n";

    my $con_ok = connect( TCP_SOCK, $destination )
        or warn "tcp_c1: connect to remote system failed: $!\n";

    next unless $con_ok;

    my $data = $msg_count . ' -> ' . $big_chunk;

    warn "Sending $msg_count to server ...\n";

    send( TCP_SOCK, $data, 0 )
        or warn "tcp_c1: problem with send: $!\n";

    sleep( 1 );
```

The First TCP Client - 3 of 3

```
    my $from_who = recv( TCP_SOCK, $data, MAX_RECV_LEN, 0 );

    if ( $from_who )
    {
        my ( $the_port, $the_ip ) = sockaddr_in( $destination );

        my $remote_name = gethostbyaddr( $the_ip, AF_INET );

        warn "Received from $remote_name: ", length( $data ),
            ' -> ', substr( $data, 0, 39 ), "\n";
    }
    else
    {
        warn "tcp_c1: problem with recv: $!\n";
    }

    close TCP_SOCK
        or warn "tcp_c1: close failed: $!\n";
}
continue {
    $msg_count++;
}
```

The First TCP Client/Server - Output: Server

```
Server starting up on port: 4001.
Received from linux303.itcarlow.ie: 1448 -> 1 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 2 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 3 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 4 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 5 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 6 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 7 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 8 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 9 -> xxxxxxxxxxxxx
Sending back to client ...
Received from linux303.itcarlow.ie: 1448 -> 10 -> xxxxxxxxxxxx
Sending back to client ...
```

The First TCP Client/Server - Output: Client

```
Sending 1 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 1 -> xxxxxxxxxxxxxx
Sending 2 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 2 -> xxxxxxxxxxxxxx
Sending 3 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 3 -> xxxxxxxxxxxxxx
Sending 4 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 4 -> xxxxxxxxxxxxxx
Sending 5 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 5 -> xxxxxxxxxxxxxx
Sending 6 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 6 -> xxxxxxxxxxxxxx
Sending 7 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 7 -> xxxxxxxxxxxxxx
Sending 8 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 8 -> xxxxxxxxxxxxxx
Sending 9 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 9 -> xxxxxxxxxxxxxx
Sending 10 to server ...
Received from pblinux.itcarlow.ie: 1448 -> 10 -> xxxxxxxxxxxxx
```

TCP Programming - Reading Everything

```
$data = '';

my $chunk;

recv( TCP_SOCK, $chunk, MAX_RECV_LEN, 0 );

while ( $chunk )
{
    $data = $data . $chunk;
    recv( TCP_SOCK, $chunk, MAX_RECV_LEN, 0 );
}
```

The Second TCP Client/Server - Output: Server

```
Server starting up on port: 4001.
Received from linux303.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 3 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 4 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 5 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 6 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 7 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 8 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65005 -> 9 -> xxxxxxxxxxxx
Sending 65005 back to client ...
Received from linux303.itcarlow.ie: 65006 -> 10 -> xxxxxxxxxxx
Sending 65006 back to client ...
```

The Second TCP Client/Server - Output: Client

```
Sending 1 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 1 -> xxxxxxxxxxxxx
Sending 2 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 2 -> xxxxxxxxxxxxx
Sending 3 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 3 -> xxxxxxxxxxxxx
Sending 4 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 4 -> xxxxxxxxxxxxx
Sending 5 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 5 -> xxxxxxxxxxxxx
Sending 6 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 6 -> xxxxxxxxxxxxx
Sending 7 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 7 -> xxxxxxxxxxxxx
Sending 8 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 8 -> xxxxxxxxxxxxx
Sending 9 65005 to server ...
Received from pblinux.itcarlow.ie: 65005 -> 9 -> xxxxxxxxxxxxx
Sending 10 65006 to server ...
Received from pblinux.itcarlow.ie: 65006 -> 10 -> xxxxxxxxxxxx
```

A Common TCP Gotcha, The Server - 1 of 2

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_TCP_PORT => 4001;

my $local_port = shift || SIMPLE_TCP_PORT;
my $trans_serv = getprotobyname( 'tcp' );
my $local_addr = sockaddr_in( $local_port, INADDR_ANY );

socket( TCP_SOCK, PF_INET, SOCK_STREAM, $trans_serv )
    or die "tcp_s2: socket creation failed: $!\n";
setsockopt( TCP_SOCK, SOL_SOCKET, SO_REUSEADDR, 1 )
    or warn "tcp_s2: could not set socket option: $!\n";
bind( TCP_SOCK, $local_addr )
    or die "tcp_s2: bind to address failed: $!\n";
listen( TCP_SOCK, SOMAXCONN )
    or die "tcp_s2: listen couldn't: $!\n";
```

A Common TCP Gotcha, The Server - 2 of 2

```
warn "Server starting up on port: $local_port.\n";

while ( accept( CLIENT_SOCK, TCP_SOCK ) )
{
    my $secs = 10;

    print CLIENT_SOCK "Sleeping for $secs seconds ... \n";
    sleep( $secs );
    print CLIENT_SOCK "I awake, only to die ... \n";

    close( CLIENT_SOCK )
        or warn "tcp_c1: close failed: $!\n";
}
close TCP_SOCK;
```

A Common TCP Gotcha, The Client - 1 of 2

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_TCP_PORT => 4001;
use constant REMOTE_HOST     => 'localhost';

my $remote = shift || REMOTE_HOST;
my $remote_port = shift || SIMPLE_TCP_PORT;
my $trans_serv  = getprotobyname( 'tcp' );
my $remote_host = gethostbyname( $remote )
    or die "tcp_c2: name lookup failed: $remote\n";
my $destination = sockaddr_in( $remote_port, $remote_host );
```

A Common TCP Gotcha, The Client - 2 of 2

```
socket( TCP_SOCK, PF_INET, SOCK_STREAM, $trans_serv )
    or die "tcp_c2: socket creation failed: $!\n";
connect( TCP_SOCK, $destination )
    or die "tcp_c2: connect to remote system failed: $!\n";

while ( <TCP_SOCK> )
{
    print $_;
}
close( TCP_SOCK );
```

A Common TCP Gotcha - Output Generated

```
Sleeping for 10 seconds ...
I awake, only to die ...
```

A Common TCP Gotcha - NetDebug Capture

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
149.153.103.5 -> 149.153.100.65 (id: 28377, ttl: 63)

TCP Source: 4001 -> TCP Destination: 1269
TCP Header Length: 8, TCP Checksum: 65123
TCP Data:

Sleeping for 10 seconds ...
I awake, only to die ...
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

A Common TCP Gotcha - The Flushing Fix

```
my $previous = select CLIENT_SOCK;
$| = 1;
select $previous;
```

A Common TCP Gotcha - Flushed Capture

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
149.153.103.5 -> 149.153.100.65 (id: 28389, ttl: 63)

TCP Source: 4001 -> TCP Destination: 1270
TCP Header Length: 8, TCP Checksum: 55511
TCP Data:

Sleeping for 10 seconds ...
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
149.153.100.65 -> 149.153.103.5 (id: 14187, ttl: 64)

TCP Source: 1270 -> TCP Destination: 4001
TCP Header Length: 8, TCP Checksum: 28529
TCP Data:
    ...
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
149.153.103.5 -> 149.153.100.65 (id: 28390, ttl: 63)

TCP Source: 4001 -> TCP Destination: 1270
TCP Header Length: 8, TCP Checksum: 34174
TCP Data:

I awake, only to die ...
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

The Remote Syntax Checker Server - 1 of 5

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_TCP_PORT => 4001;

my $local_port = shift || SIMPLE_TCP_PORT;
my $trans_serv = getprotobyname( 'tcp' );
my $local_addr = sockaddr_in( $local_port, INADDR_ANY );

socket( TCP_SOCK, PF_INET, SOCK_STREAM, $trans_serv )
    or die "tcp_s4: socket creation failed: $!\n";
setsockopt( TCP_SOCK, SOL_SOCKET, SO_REUSEADDR, 1 )
    or warn "tcp_s4: could not set socket option: $!\n";
bind( TCP_SOCK, $local_addr )
    or die "tcp_s4: bind to address failed: $!\n";
listen( TCP_SOCK, SOMAXCONN )
    or die "tcp_s4: listen couldn't: $!\n";

warn "Server starting up on port: $local_port.\n";
```

The Remote Syntax Checker Server - 2 of 5

```
my $from_who;

while ( $from_who = accept( CLIENT_SOCK, TCP_SOCK ) )
{
    my $data = '';

    my $previous = select CLIENT_SOCK;
    $| = 1;
    select $previous;

    my $tmp_fn = <CLIENT_SOCK>;

    chomp( $tmp_fn );

    $tmp_fn = ( split /\//, $tmp_fn )[-1];
```

The Remote Syntax Checker Server - 3 of 5

```
while ( my $chunk = <CLIENT_SOCK> )
{
    $data = $data . $chunk;
}

open FILETOCHECK, ">$tmp_fn";

print FILETOCHECK $data;

close FILETOCHECK;
```

The Remote Syntax Checker Server - 4 of 5

```
my $cmd = "perl -c $tmp_fn 2>&1 1>/dev/null";

my ( $the_port, $the_ip ) = sockaddr_in( $from_who );

my $remote_name = gethostbyaddr( $the_ip, AF_INET )
    || inet_ntoa( $the_ip );

warn "tcp_s4: executing: [ $cmd ] for $remote_name\n";

my $stderr_output = qx/$cmd/;
```

The Remote Syntax Checker Server - 5 of 5

```
    print CLIENT_SOCK $stderr_output;

    close CLIENT_SOCK
        or warn "tcp_s4: close failed: $!\n";

    unlink $tmp_fn;
}

close TCP_SOCK;
```

The Remote Syntax Checker Client - 1 of 5

```
#! /usr/bin/perl -w

use strict;
use Socket;

use constant SIMPLE_TCP_PORT => 4001;
use constant REMOTE_HOST     => 'localhost';
```

The Remote Syntax Checker Client - 2 of 5

```
my $filetosend = shift;

die "tcp_c3: you must provide a filename: $!\n"
    unless defined $filetosend;

open TOSEND, "$filetosend"
    or die "tcp_c3: could not open $filetosend: $!\n";

my @entire_file = <TOSEND>;

close TOSEND;
```

The Remote Syntax Checker Client - 3 of 5

```
my $remote = shift || REMOTE_HOST;
my $remote_port = shift || SIMPLE_TCP_PORT;

my $trans_serv  = getprotobyname( 'tcp' );
my $remote_host = gethostbyname( $remote )
    or die "tcp_c3: name lookup failed: $remote\n";
my $destination = sockaddr_in( $remote_port, $remote_host );

socket( TCP_SOCK, PF_INET, SOCK_STREAM, $trans_serv )
    or die "tcp_c3: socket creation failed: $!\n";
connect( TCP_SOCK, $destination )
    or die "tcp_c3: connect to remote system failed: $!\n";

my $previous = select TCP_SOCK;
$| = 1;
select $previous;
```

The Remote Syntax Checker Client - 4 of 5

```
print TCP_SOCK $filetosend . "\n";

print TCP_SOCK @entire_file;

shutdown( TCP_SOCK, 1 );
```

The Remote Syntax Checker Client - 5 of 5

```
my $data = '';

while ( my $chunk = <TCP_SOCK> )
{
    $data = $data . $chunk;
}

close TCP_SOCK
    or warn "tcp_c3: close failed: $!\n";

print "$data";
```

The Remote Syntax Checker Server Messages

```
Server starting up on port: 4001.
tcp_s4: executing: [ perl -c hosts 2>&1 1>/dev/null ]
    for 149.153.103.15
tcp_s4: executing: [ perl -c udp_c8 2>&1 1>/dev/null ]
    for 149.153.103.15
tcp_s4: executing: [ perl -c udp_c1 2>&1 1>/dev/null ]
    for pblinux.itcarlow.ie
tcp_s4: executing: [ perl -c tcp_s5 2>&1 1>/dev/null ]
    for pblinux.itcarlow.ie
```

The Remote Syntax Checker Client
Example Output - 1 of 3

```
./tcp_c3 /etc/hosts linux303

Bareword found where operator expected at hosts line 1,
near "0.1localhost" (Missing operator before localhost?)
syntax error at hosts line 1, near "0.1 localhostlocalhost"
Number found where operator expected at hosts line 2,
near "149.153" (Missing semicolon on previous line?)
Bareword found where operator expected at hosts line 2,
near "103.15 linux303_15" (Missing operator before linux303_15?)
hosts had compilation errors.
```

The Remote Syntax Checker Client Example Output - 2 of 3

```
./tcp_c3 udp_c8 linux303

udp_c8 syntax OK
```

The Remote Syntax Checker Client Example Output - 3 of 3

```
./tcp_c3 tcp_s5 linux303

Bareword "CLIENT_SOCK" not allowed while "strict subs" in
use at tcp_s5 line 86.
tcp_s5 had compilation errors.
```

The Concurrent Syntax Checker Server - 1 of 7

```
#! /usr/bin/perl -w

use strict;
use Socket;

use POSIX ":sys_wait_h";

sub zombie_reaper {
    while ( waitpid( -1, WNOHANG ) > 0 )
    { }
    $SIG{CHLD} = \&zombie_reaper;
}

$SIG{CHLD} = \&zombie_reaper;
```

The Concurrent Syntax Checker Server - 2 of 7

```
sub continue_as_child {
    my $handle = shift;
    my $from_who_client = shift;

    my $data = '';

    my $previous = select $handle;
    $| = 1;
    select $previous;

    my $tmp_fn = <$handle>;

    chomp( $tmp_fn );

    $tmp_fn = (split /\//, $tmp_fn)[-1];
```

The Concurrent Syntax Checker Server - 3 of 7

```
while ( my $chunk = <$handle> )
{
    $data = $data . $chunk;
}

open FILETOCHECK, ">$tmp_fn";

print FILETOCHECK $data;

close FILETOCHECK;

my $cmd = "perl -c $tmp_fn 2>&1 1>/dev/null";
```

The Concurrent Syntax Checker Server - 4 of 7

```
    my ( $the_port, $the_ip ) = sockaddr_in( $from_who_client );

    my $remote_name = gethostbyaddr( $the_ip, AF_INET )
        || inet_ntoa( $the_ip );

    warn "tcp_s5: executing: [ $cmd ] for $remote_name\n";

    my $stderr_output = qx( $cmd );

    print $handle $stderr_output;

    close $handle
        or warn "tcp_s5: close failed: $!\n";

    unlink $tmp_fn;
}
```

The Concurrent Syntax Checker Server - 5 of 7

```
use constant SIMPLE_TCP_PORT => 4001;

my $local_port = shift || SIMPLE_TCP_PORT;
my $trans_serv = getprotobyname( 'tcp' );
my $local_addr = sockaddr_in( $local_port, INADDR_ANY );

socket( TCP_SOCK, PF_INET, SOCK_STREAM, $trans_serv )
    or die "tcp_s5: socket creation failed: $!\n";
setsockopt( TCP_SOCK, SOL_SOCKET, SO_REUSEADDR, 1 )
    or warn "tcp_s5: could not set socket option: $!\n";
bind( TCP_SOCK, $local_addr )
    or die "tcp_s5: bind to address failed: $!\n";
listen( TCP_SOCK, SOMAXCONN )
    or die "tcp_s5: listen couldn't: $!\n";

warn "Server starting up on port: $local_port.\n";
```

The Concurrent Syntax Checker Server - 6 of 7

```
my $from_who;

while ( $from_who = accept( CLIENT_SOCK, TCP_SOCK ) )
{
    my $child_pid = fork;

    if ( $child_pid )
    {
        next;
    }
    elsif ( defined( $child_pid ) )
    {
        close( TCP_SOCK );
        continue_as_child( *CLIENT_SOCK, $from_who );
        exit;
    }
```

The Concurrent Syntax Checker Server - 7 of 7

```
    else
    {
        warn "tcp_s5: fork failed: $!\n";
    }
}
continue {
    close( CLIENT_SOCK );
}

close TCP_SOCK;
```

OO Sockets with IO::Socket::INET

```
my $sock_obj = IO::Socket::INET->new( LocalPort   => 4001,
                                      Proto       => 'tcp',
                                      Listen      => SOMAXCONN )
    or die "Could not create socket object: $!\n";

    ...

my $connected_client_obj = $sock_obj->accept
    or die "Unable to accept a connection: $!\n";

my $data;

$connected_client_obj->recv( $data, MAX_RECV_LEN, 0 );
$connected_client_obj->send( $data, 0 );

    ...

my $data;

$data = <$connected_client_obj>;
print $connected_client_obj $data;
```

An Object Oriented Server - 1 of 2

```
#! /usr/bin/perl -w

use strict;
use IO::Socket;

use constant SIMPLE_TCP_PORT => 4001;

my $port = shift || SIMPLE_TCP_PORT;

my $sock_obj = IO::Socket::INET->new( LocalPort   => $port,
                                      Proto       => 'tcp',
                                      Reuse       => 1,
                                      Listen      => SOMAXCONN )
    or die "oo_tcp_s3: could not create socket object: $!\n";

warn "OO Server starting up on port: ", $sock_obj->sockport, ".\n";
```

An Object Oriented Server - 2 of 2

```
while ( my $client_obj = $sock_obj->accept )
{
    my $secs = 10;

    print $client_obj "Sleeping for $secs seconds ... \n";
    sleep( $secs );
    print $client_obj "I awake, only to die ... \n";

    $client_obj->close
        or warn "oo_tcp_s3: close failed: $!\n";
}
$sock_obj->close;
```

An Object Oriented Client

```
#! /usr/bin/perl -w

use strict;
use IO::Socket;

use constant SIMPLE_TCP_PORT => 4001;
use constant REMOTE_HOST     => 'localhost';

my $remote = shift || REMOTE_HOST;
my $remote_port = shift || SIMPLE_TCP_PORT;

my $sock_obj = IO::Socket::INET->new( PeerAddr => $remote,
                                      PeerPort => $remote_port,
                                      Proto    => 'tcp' )
    or die "oo_tcp_c3: could not create socket object: $!\n";

while ( <$sock_obj> )
{
    print $_;
}
$sock_obj->close;
```