Cliente.pl

```
#!/usr/local/bin/perl
use Socket;
($them, $port) = @ARGV;
```

El primer argumento es el nombre de la máquina a la que conectamos «them», y el siguiente, el puerto «port»

```
$port = 2345 unless $port;
$them = 'localhost' unless $them;
```

Si no se da nombre de host, se asume que es «localhost»

Cliente.pl (2)

```
$SIG{'Int'} = 'dokill';
```

El vector de interrupciones es una tabla hash «SIG»

Programamos «Int» para llamar a «dokill»

```
sub dokill {
    kill 9, $child if $child;
}
```

La variable «child» nos ayuda a hacer la limpieza, dado que necesitaremos un proceso hijo.

Averiguamos el nombre local «`hostname`»

y eliminamos el retorno de carro de la línea

```
chop($hostname = `hostname`);
```

Cliente.pl (3)

```
($name, $aliases, $proto) = getprotobyname('tcp');
($name, $aliases, $port) = getservbyname($port,'tcp')
    unless $port =~ /^\d+$/;

print "Using port $port to connect to server on host
    $them...\n";
($name,$aliases,$type,$len,$thisaddr) =
    gethostbyname($hostname);
($name, $aliases,$type,$len,$thataddr) =
    gethostbyname($them);
```

Ligeramente diferente al del servidor, pero con idéntico efecto.

nuestra IP a partir del nombre del host.

su IP a partir de «them».

Cliente.pl (4)

```
if (socket(S,AF_INET, SOCK_STREAM, $proto)) {
    print "Socket creation succeeded.\n";
} else {    die $!;  }

$sockaddr = 'S n a4 x8';
$this = pack($sockaddr, AF_INET, 0, $thisaddr);
$that = pack($sockaddr, AF_INET, $port, $thataddr);
```

Ligeramente diferente al del servidor, pero con idéntico efecto.

Empaqueta las direcciones para el *binding*

Cliente.pl (5)

```
if (bind(S, $this)) {
    print "Bind succeeded.\n";
}  else  {   die $!; }

if (connect(S, $that)) {
    print "Connect succeeded.\n";
}  else {    die $!; }

select(S);
$| = 1;
select(STDOUT);
```

«connect» inicia la conexión de los sockets.

No hay «listen»

Cliente.pl (6)

```
if ($child = fork) {
    while (<STDIN>) {
        print S;
    }

    # Sleep for 3 seconds then...
    sleep 3;
    #...then kill ourselves and the child
    do dokill();
}
```

«fork»:

- un proceso leerá de la entrada estándar y envía por el socket, y
- otro lee del socket e imprime en la salida estándar.

aquí acaba el código del hijo:

ojo: es el fín del stream del hijo el que para también al padre

Cliente.pl (7)

```
    else {

        while (<S>) {
        print "Server: $_";
    }
}
```

El padre finalizará cuando llegue el final del stream del servidor, pero esto **no debiera ocurrir**.

OJO: para acabar, finalizaremos con ^D el proceso cliente, y luego los servidores.