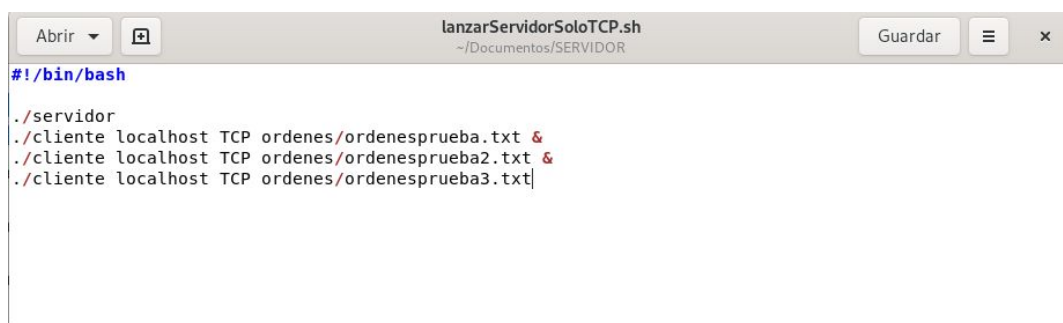


Nombre y apellidos	DNI
Erick José Mercado Hernandez	43840208-T
Javier Merchán Benito	70923644-Q

Conexión cliente-servidor NNTP TCP


En las conexiones TCP, primeramente el servidor debe crear el socket y posteriormente quedarse escuchando hasta que el cliente se conecte a él. Una vez el cliente ya se ha conectado se producirá el intercambio de mensajes entre ellos.

En nuestro caso, se ejecutará el servidor como un daemon creando un proceso diferente para atender a cada cliente que solicite conexión con el servidor de forma individual, pudiendo demostrar que pueden estar más de un solo cliente conectado.



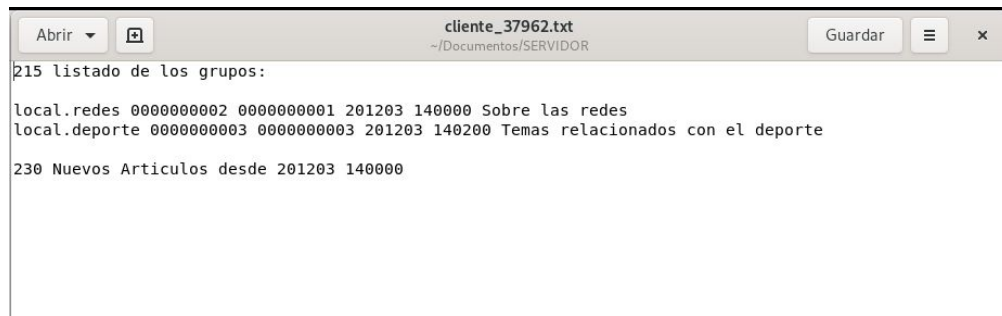
```
#!/bin/bash
./servidor
./cliente localhost TCP ordenes/ordenesprueba.txt &
./cliente localhost TCP ordenes/ordenesprueba2.txt &
./cliente localhost TCP ordenes/ordenesprueba3.txt|
```

- Ficheros .txt de ejecución:



```
list
NEWNEWS local.deporte 201203 140000
quit|
```

Como se puede observar, en el primer fichero de ejecución vamos a mostrar list, newnews y quit, tres de las órdenes de NNTP que había que codificar.



```
cliente_37962.txt
~/Documentos/SERVIDOR

215 listado de los grupos:

local.redes 0000000002 0000000001 201203 140000 Sobre las redes
local.deporte 0000000003 0000000003 201203 140200 Temas relacionados con el deporte

230 Nuevos Articulos desde 201203 140000
```

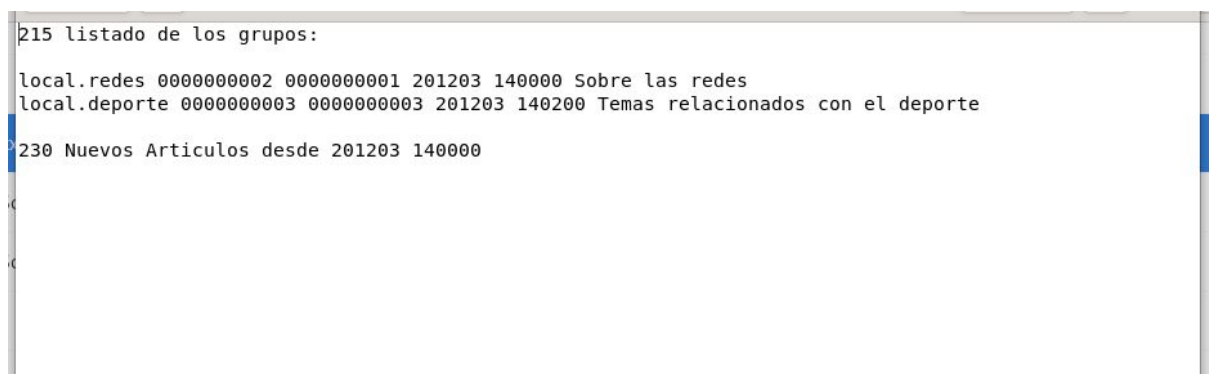
El resultado de la ejecución de lo anterior se guarda en un fichero .txt con el número del puerto efímero del cliente. Newnews no imprime todo en el fichero pero sí que realiza la transferencia de datos correctamente, ya que por printf sí que muestra correctamente todo el contenido.



```
ordenesprueba2.txt
~/Documentos/SERVIDOR/ordenes

list
NEWNEWS local.redes 201203 140000
NEWGROUPS 201203 140000
quit
```

En el segundo fichero de ejecución vamos a mostrar list, newnews, newgroups y quit.



```
215 listado de los grupos:

local.redes 0000000002 0000000001 201203 140000 Sobre las redes
local.deporte 0000000003 0000000003 201203 140200 Temas relacionados con el deporte

230 Nuevos Articulos desde 201203 140000
```



En el tercer fichero, tenemos la ejecución de post que debe realizarse sin que el cliente se ejecute en segundo plano ya que no hemos conseguido que sea capaz de leer la cabecera y el cuerpo desde el fichero.

```
post
340 Subiendo un articulo; finalize con una linea que solo contenga un .
NEWSGROUPS: local.test
noticias/articulos/local/test

2
SUBJECT: esto es una prueba
BODY:

cuerpo de articulo de prueba
.
240 Artículo Recibido correctamente
quit
Servidor: 205 adios
```

```
NEWSGROUPS: local.test

SUBJECT: esto es una prueba
DATE: 210110 104515 Sun Jan 10 10:45:15 2021
Message-ID: <1@nagal.usal.es>

cuerpo de articulo de prueba
.
```

Viendo los resultados de ejecución, se observa como desde ordenesprueba3.txt entra en la orden post y desde teclado se introducen los valores de los diferentes campos que componen la orden. También se adjunta una imagen del texto creado en la carpeta de noticias que irá en el .zip

El servidor registra todas las conexiones y órdenes introducidas en un .log del cual se muestra unas imagenes a continuación:

```

nntp.log
~/Documentos/SERVIDOR
Guardar

Startup from localhost port 37962 client_IP 32512 at Sun Jan 10 10:38:08 2021
Startup from localhost port 37964 client_IP 32512 at Sun Jan 10 10:38:08 2021
Startup from localhost port 37966 client_IP 32512 at Sun Jan 10 10:38:08 2021

210110 103808 TCPClient: list
, localhost, 37966
210110 103808 TCPClient: list
, localhost, 37962
210110 103808 TCPClient: list
, localhost, 37964
215 listado de los grupos:

local.redes 0000000002 0000000001 201203 140000 Sobre las redes
local.deporte 0000000003 0000000003 201203 140200 Temas relacionados con el deporte

215 listado de los grupos:

local.redes 0000000002 0000000001 201203 140000 Sobre las redes
local.deporte 0000000003 0000000003 201203 140200 Temas relacionados con el deporte

210110 103808 TCPClient: NEWNEWS local.deporte 201203 140000
, localhost, 37962
210110 103808 TCPClient: NEWNEWS local.redes 201203 140000
, localhost, 37966
215 listado de los grupos:

local.redes 0000000002 0000000001 201203 140000 Sobre las redes
local.deporte 0000000003 0000000003 201203 140200 Temas relacionados con el deporte

210110 103808 TCPClient: NEWGROUPS 201130 000000
, localhost, 37964

210110 103808 TCPClient: quit
, localhost, 37964
205 adios

210110 103808 TCPClient: , localhost, 37962 ,500 comando desconocido

210110 103808 TCPClient: , localhost, 37962 ,500 comando desconocido

210110 103808 TCPClient: , localhost, 37966 ,500 comando desconocido
```

```
Startup from localhost port 37968 client_IP 32512 at Sun Jan 10 10:44:08 2021
210110 104408 TCPClient: , localhost, 37968 ,500 comando desconocido
210110 104408 TCPClient: , localhost, 37968 ,500 comando desconocido
Startup from localhost port 37970 client_IP 32512 at Sun Jan 10 10:45:00 2021
210110 104500 TCPClient: list
, localhost, 37970
215 listado de los grupos:
local.redes 0000000002 0000000001 201203 140000 Sobre las redes
local.deporte 0000000003 0000000003 201203 140200 Temas relacionados con el deporte
210110 104500 TCPClient: NEWNEWS local.deporte 201203 140000
, localhost, 37970
210110 104500 TCPClient: , localhost, 37970 ,500 comando desconocido
210110 104500 TCPClient: , localhost, 37970 ,500 comando desconocido
Startup from localhost port 37972 client_IP 32512 at Sun Jan 10 10:45:15 2021
210110 104515 TCPClient: post
, localhost, 37972
210110 104515 TCPClient: quit
, localhost, 37972
205 adios
```

Debido a el problema expuesto en las siguientes imágenes, no hemos podido ejecutar de forma correcta las órdenes que tienen que ver con la orden group, por algún motivo que que desconocemos se solapa, solapandose los dos últimos caracteres con los dos primeros, el valor de los buffers al final cuando tiene un tamaño más que suficiente y la orden está introducida dentro de un sprintf con todo en orden, incluso indicando que en el final de la cadena se introduce un \0, este error empezó a ocurrir cuando se leía de fichero, anteriormente la ejecución era correcta.

```
a3.txt
group
article 1
  Ingrese por : 4
  El Valor de token2 es : redes
  El Valor de token es : article
  El Valor de token es : 1

  El Valor de token3 es : 1
article
article 1

/lticias/articulos/local/redes

Servidor: 500 comando desconocido

quit
```

```
211 numeros de articulos primero y ultimo de: local.redes  
Numero de articulos 2 Ultimo 0000000002 primero: 0000000001  
423 no existe este articulo en el grupo de noticias
```

Como se observa en esta ejecución sacada en uno de los .txt de los clientes, la orden group se imprime correctamente pero cuando se introduce las órdenes article, head o body ocurre el error antes expuesto y no encuentra correctamente el artículo. La comunicación ocurre correctamente. Si se introduce la dirección completa como una cadena de caracteres (char articulo[MAX]="noticias/articulos/local/redes/1");, el article, head o body recupera bien mostrando el mensaje "223, 222 o 221" de que ha tenido éxito y el contenido correspondiente.

Adjuntamos imagen del nntp.log del servidor con el error:

```
Startup from localhost port 37978 client_IP 32512 at Sun Jan 10 10:52:48 2021  
  
210110 105248 TCPClient:group local.redes  
, localhost, 37978  
211 numeros de articulos primero y ultimo de: local.redes 0000000002 0000000001 201203 140000  
Sobre las redes  
Numero de articulos 2 Ultimo 0000000002 primero: 0000000001  
210110 105248 TCPClient:article 1  
, localhost, 37978  
423 no existe este articulo en el grupo de noticias  
  
210110 105248 TCPClient: noticias/articulos/local/redes  
/1, localhost, 37978 ,500 comando desconocido  
  
210110 105248 TCPClient: quit  
, localhost, 37978  
205 adios
```

Conexión cliente-servidor NNTP UDP

En las conexiones UDP, el servidor y el cliente no tiene que estar conectados el uno con el otro, el servidor se queda bloqueado en el `receivefrom` hasta que algún cliente introduce alguna orden y se produce la comunicación.

En nuestro caso, no hemos conseguido que el cliente lea del fichero las órdenes debido a que ignora las órdenes que le introducimos o al ejecutar el cliente en segundo plano este se muere antes de conectar con el servidor. Por todo ello adjuntamos un “.sh” único para el clientes en UDP y todas las órdenes que deba realizar se hacen por consola.



```
Abrir ▾ [icon] lanzarServidorSoloUDP.sh [icon] Guardar [icon] x
~/Documentos/SERVIDOR
#!/bin/bash
./servidor
./cliente localhost UDP vacio
```

Cuando se ejecuta la orden `list` desde el terminal se observa cómo se produce la comunicación con el servidor y le devuelve al cliente lo que ha solicitado:



```
Introduzca una orden : list
Client : list

Servidor :
Servidor : 215 listado de los grupos:
Servidor : local.redes 0000000002 0000000001 201203 140000 Sobre las redes
local.deporte 0000000003 0000000003 201203 140200 Temas relacionados con el depo
rte
```

Ejecucion de newgroups desde terminal:


```
Client : NEWGROUPS 201203 140000
231 listado de grupos desde: 201203 140000
Servidor : 231 listado de grupos desde: 201203 140000
Servidor : local.redes
local.deporte
Introduzca una orden : █
Introduzca una orden : group local.deporte
Servidor : 211 numeros de articulos primero y ultimo del grupo local.deporte
Servidor : Numero de articulos 1 Ultimo 0000000003 primero: 0000000003
local.deporte
```

Viendo los resultados de la ejecución en el terminal, se puede observar como se muestra correctamente el contenido por lo que la comunicación sería correcta.

Orden group en UDP desde terminal:

```
Introduzca una orden : group local.redes
Servidor : 211 numeros de articulos primero y ultimo del grupo local.redes
local.redes
Numero de articulos 2 Ultimo 0000000002 primero: 0000000001
Servidor : Numero de articulos 2 Ultimo 0000000002 primero: 0000000001
local.redes
█
```

Se observa como el contenido de la orden introducida es el solicitado por el cliente.

Órdenes article, head y body en UDP, que al contrario de lo que ocurría en TCP se observa como la dirección pasada a dichas órdenes si es la correcta, TCP tenía en un principio el mismo código que estas funciones de UDP pero continuaba haciendo el error ya mencionado.

Ejecución de las órdenes en UDP:

```
head 1
noticias/articulos/local/redes/1
noticias/articulos/local/redes/1
noticias/articulos/local/redes/1
lo encuentre
221 cabecera del articulo recuperada

Servidor : Newsgroups: local.redes
Servidor : Subject: Sobre los sockets
Servidor : Date: 201203 143209 Thu, 3 Dec 2020 14:32:09 -0000 (UTC)
Servidor : Message-ID: <l@nocal.usal.es>
```

```
body 1
noticias/articulos/local/redes/1
noticias/articulos/local/redes/1
noticias/articulos/local/redes/1
lo encuentre
222 cuerpo del articulo recuperado

Servidor :
Servidor : Los sockets son un API de programación en red.
Servidor : Permite utilizar los servicios de los protocolos de transporte TCP y
UDP.
Servidor : .
```

```
article 1
noticias/articulos/local/redes/1
noticias/articulos/local/redes/1
noticias/articulos/local/redes/1
lo encuentre
223 articulo recuperado

Servidor : Newsgroups: local.redes
Servidor : Subject: Sobre los sockets
Servidor : Date: 201203 143209 Thu, 3 Dec 2020 14:32:09 -0000 (UTC)
Servidor : Message-ID: <l@nocal.usal.es>
```

Como se puede ver el resultado de la ejecución de head y body son correctos, pero en article la ejecución recibe toda la información requerida pero al imprimir por pantalla el resultado solo muestra hasta la cabecera del artículo.

Orden post, que también habría que introducirla por el terminal y como ocurría en TCP habría que introducir todos los campos desde terminal hasta introducir un "." que es cuando el artículo se publicaría en la carpeta de noticias adjuntada.

```
Introduzca una orden : post
340 Subiendo un artículo; finalize con una línea que solo contenga un .
NEWSGROUPS: local.test
noticias/articulos/local/test

3
..
.
1
SUBJECT: este es una prueba de UDP
BODY:

Cuerpo texto de prueba UDP
.
240 Artículo Recibido correctamente
```

```
NEWSGROUPS: local.test

SUBJECT: este es una prueba de UDP
DATE: 210110 113615 Sun Jan 10 11:36:15 2021
Message-ID: <2@micasa>

Cuerpo texto de prueba UDP
.
```

Como se ha podido ver en las imágenes el servidor recibe correctamente todos los campos y publica el artículo.

Orden quit:

```
Introduzca una orden : quit
Servidor: 205 adios
```

Listado de archivos adjuntos en el .zip

- Carpeta servidor:
 - Servidor .c
 - Cliente .c
 - lanzarServidorSoloTCP .sh
 - lanzarServidorSoloUDP.sh
 - Compila.sh (Llama al makefile para compilar todo junto)
 - Makefile
 - Carpeta funciones: contiene todos las funciones del cliente y del servidor.

- Carpeta noticias: contiene los resultados y búsquedas.
 - Carpeta órdenes: contiene los ficheros .txt con las órdenes.
 - Carpeta clientes: en ella se guardaran los .txt con las órdenes que reciba el cliente.
- InformePracticaSockets.pdf (este pdf)