Sockets: Servicio HTTP simplificado

Versión TCP

SERVIDOR -- server.c

El funcionamiento base se encuentra entre las líneas **433 - 607**

En el fichero del Servidor de la versión TCP hay varias cosas a destacar. Al ser la versión TCP existe una conexión entre el cliente y el servidor por lo que entrará en un bucle donde irá recibiendo las peticiones del Cliente. En HTTP sólo hay dos tipos de mensajes: las peticiones del Cliente al Servidor y las Respuestas de este.

Se le quita el CR-LF al comando recibido y se va dividiendo, cogiendo las partes del comando necesarias para el correcto funcionamiento del servidor:

```
FILE *ficheroParam;
char c;
int count;
char fichero[20], *modo;
bool numParams = false;
int i,d;

char linea[TAMANO];
char *nombreComando;

nombreComando = strtok(command, " ");


strcpy(fichero,".");
strcat(fichero,strtok(NULL, " "));

modo = strtok(NULL, " ");

if(modo == NULL){
        strcpy(modo,"c");
}

if(strtok(NULL, " ") != NULL) numParams = true;

//INICIO
char inicio[100];
RESET(inicio,100);
if(numParams || strcmp(nombreComando,"GET") != 0){
        strcpy(inicio,"HTTP/1.1 501 Not Implemented");
        i = 501;
}

else if(NULL == (ficheroParam = fopen(fichero, "r"))){
        strcpy(inicio,"HTTP/1.1 404 Not Found");
        i = 404;
}else{
        strcpy(inicio,"HTTP/1.1 200 OK");
        i = 200;
}
addCRLF(inicio,strlen(inicio));
//Send 1
send(s, inicio,TAMANO, 0);
if(-1 == addCommandToLog(inicio, true)){
        perror("No se ha podido a•adir la respuesta al fichero http.log");
}
```

Una vez dividido el comando con “strtok” se van comparando las partes, primero la orden, que el que caso de que sea una distinta de GET escribirá *HTTP/1.1 501 Not Implemented*, si no consigue leer el archivo entonces: *HTTP/1.1 404 Not Found*, porque no encuentra la página y finalmente si no se cumple ninguna de las anteriores *HTTP/1.1 200 OK,* todo fue bien.

Añade el CR-LF y hace el primer envío al Cliente con lo anterior.

```
508                 //NOMBRE SERVIDOR
509                 char servidor[TAMANO];
510                 strcpy(servidor,"Server: Servidor de Javier");
511                 addCRLF(servidor,TAMANO);
512                 //Send 2
513                 send(s, servidor, TAMANO, 0);
514                 if(-1 == addCommandToLog(servidor, true)){
515                         perror("No se ha podido a?adir la respuesta al fichero http.log");
516                 }
517
518                 //CONEXION
519                 char conexion[TAMANO];
520                 if(strcmp(modo,"k") == 0){
521                         strcpy(conexion,"Connection: keep-alive");
522                 }else{
523                         strcpy(conexion,"Connection: close");
524                         finish=true;
525                 }
526                 addCRLF(conexion,TAMANO);
527                 //Send 3
528                 send(s, conexion, TAMANO, 0);
529                 if(-1 == addCommandToLog(conexion, true)){
530                         perror("No se ha podido a?adir la respuesta al fichero http.log");
531                 }
532                 //CONTENT LENGTH
533                 int longitud = 0;
534                 char content[TAMANO];
535                 strcpy(content,"Content-Length: ");
536                 if(i == 200){
537                         fseek(ficheroParam, 0, SEEK_END);
538                         longitud = ftell(ficheroParam);
539                         count = numLines(ficheroParam);
540                         longitud = longitud - count;
541                         char l[5];
542                         sprintf(l, "%d", longitud);
543                         strcat(content,l);
544                 }else if(i == 501 || i == 404){
545                         char l[5];
546                         longitud = 60;
547                         sprintf(l, "%d", longitud);
548                         strcat(content,l);
549                 }
```

Después sigue con los siguientes mensajes y escribiendo en el http.log, el nombre del servidor y hace el envío al cliente, el tipo de conexión, si es **k** significa que la conexión con el servidor seguirá activa, si es **c** o en blanco la conexión terminará y si el cliente envía más mensajes nunca llegarán al servidor hasta que no vuelva a inicializar la conexión. El cuerpo del Mensaje con la cabecera Content Length, terminando la cabecera con el CR-LF

```
//Send 4
sendto(s, content, TAMANO, 0,(struct sockaddr *) &clientaddr_in, addrlen);
if(-1 == addCommandToLog(content, true)){
        perror("No se ha podido a♦adir la respuesta al fichero http.log");
}
//VACIA
char vacia[TAMANO];
strcpy(vacia,"");
addCRLF(vacia,strlen(vacia));
//Send 5
sendto(s, vacia, TAMANO, 0,(struct sockaddr *) &clientaddr_in, addrlen);
if(-1 == addCommandToLog(vacia, true)){
        perror("No se ha podido a♦adir la respuesta al fichero http.log");
}

char texto[longitud];
if(i == 200){
        fseek(ficheroParam, 0, SEEK_SET);
        while( fgets(linea, 200, ficheroParam) ){
                linea[strlen(linea) - 1] = '\0';
                strcat(texto,linea);
        }
        fclose(ficheroParam);
}else if(i == 501){
        strcpy(texto,"<html><body><h1> 501 Not Implemented </h1></body></html>");
}else if(i == 404){
        strcpy(texto,"<html><body><h1> 404 Not Implemented </h1></body></html>");
}
//Send 6
sendto(s, texto, TAMANO, 0,(struct sockaddr *) &clientaddr_in, addrlen);
if(-1 == addCommandToLog(texto, true)){
        perror("No se ha podido a♦adir la respuesta al fichero http.log");
}


if(finish) break;
```

Y finalmente el cuerpo del mensaje dependiendo de la orden que se tomará al principio se envía al cliente y posteriormente en el http.log.

CLIENTE

Tiene su propio archivo client_tcp.c. Para acceder a ella src-->client-->clients

El cliente se conecta al servidor a través del puerto usando “htons” (Host to Ntework Short).

```c
/* PORT del servidor en orden de red*/
    servaddr_in.sin_port = htons(PORT);

    if (connect(s, (const struct sockaddr *)&servaddr_in, sizeof(struct sockaddr_in)) == -1) {
            perror(argv[0]);
            fprintf(stderr, "%s: unable to connect to remote\n", argv[0]);
            exit(1);
    }
            /
    addrlen = sizeof(struct sockaddr_in);
    if (getsockname(s, (struct sockaddr *)&myaddr_in, &addrlen) == -1) {
            perror(argv[0]);
            fprintf(stderr, "%s: unable to read socket address\n", argv[0]);
            exit(1);
    }
    time(&timevar);

    sprintf(clientLogName, "../bin/logs/%u.txt", ntohs(myaddr_in.sin_port));
    if(NULL == (clientLog = fopen(clientLogName, "a+"))){
            perror("Error al iniciar el fichero .txt del cliente");
            exit(1);
    }

    fprintf(clientLog,"[TCP] Connected to %s on port %u at %s\n",
                    argv[1], ntohs(myaddr_in.sin_port), (char *) ctime(&timevar));
    fclose(clientLog);
    if(NULL == (clientLog = fopen(clientLogName, "a+"))){
            perror("Error al iniciar el fichero .txt del cliente");
            exit(1);
    }
```

Se conecta con su puerto efímero y escribe en su archivo de registro que se ha conectado correctamente al servidor x con puerto efímero **z** y en la fecha y hora determinada.

Empieza a leer el Archivo de Órdenes:

```c
//AQUI CADA VEZ QUE ENTRA LEE UNA LINEA
RESET(command, TAMANO);
while( fgets(command, sizeof(command), commandsFile) != NULL){
        command[strlen(command)-2] = '\0';

        if(command[0] == '\0')
                continue;

        addCRLF(command, TAMANO);
        //Envia comando al server
        if (send(s, command, TAMANO, 0) != TAMANO) {
                fprintf(stderr, "%s: Connection aborted on error ", argv[0]);
                fprintf(stderr, "on send number %d\n", i);
                exit(1);
        }

        if(removeCRLF(command)){
                fprintf(stderr, "[TCP] Command without CR-LF. Aborted \"connection\" \n");
                exit(1);
        }

        fprintf(clientLog, "C: %s\n", command);
        fclose(clientLog);
        if(NULL == (clientLog = fopen(clientLogName, "a+"))){
                perror("Error al iniciar el fichero .txt del cliente");
                exit(1);
        }
```

Le añade el CR-LF y envía la primera petición al Servidor, el comando leído del archivo, que éste recibirá y empezará a gestionar. Después se vuelve a quitar el CR-LF y se escribe en el archivo del cliente el comando.

Una vez hecho esto empiezan las llegadas, respuestas del servidor, tienen que haber 6 respuestas del servidor como estás. En las que el Cliente recibe a través de la función "recvTCP" la respuesta que le haya enviado el Servidor, le quita el CR-LF y la escribe en el archivo del cliente. Así con las 5 siguientes.

```
//Comienzan las llegadas
RESET(response, TAMANO);
//Received response 1
if(-1 == recvTCP(s, response, TAMANO)){
        perror(argv[0]);
        fprintf(stderr, "[TCP] %s: error reading result 1\n", argv[0]);
        exit(1);
}
if(removeCRLF(response)){
        fprintf(stderr, "[TCP] Response without CR-LF. Aborted connection\n");
        exit(1);
}
fprintf(clientLog, "S: %s\n", response);
fclose(clientLog);
if(NULL == (clientLog = fopen(clientLogName, "a+"))){
        perror("Error al iniciar el fichero .txt del cliente");
        exit(1);
}
```

Cuando termina, lee la siguiente orden del fichero y empieza de nuevo el intercambio de mensajes entre el Cliente y el Servidor, hasta que termine el archivo de comandos y entonces lo cierre y termine la conexión, ya que el cliente no le va a enviar más peticiones al servidor.

Versión UDP

El archivo server.c

SERVIDOR

El funcionamiento base se encuentra entre las líneas **690 - 854**

En cuanto a la función serverUDP simplemente destacar que el envío de mensajes se realiza mediante la función sendTo, en vez de send como era anteriormente. La variable booleana finish que sirve para finalizar la conexión (salir del bucle while) siempre es True.

Esto es debido a que el servidor en UDP está en modo escucha con un socket en dicho modo. Ese socket se encargará de cada vez que el cliente quiera enviar un comando al servidor, éste creará un nuevo socket por el cual el cliente tendrá una conexión privada. Por eso cada vez que el cliente mande un comando establecerá conexión con un nuevo socket.

Por lo demás sigue siendo similar a la versión de TCP ya explicada.

CLIENTE

Tiene su propio archivo client_udp.c. Para acceder a ella src-->client-->clients

En cuanto al cliente podemos observar el cambio más importante:

```
168         while( fgets(command, sizeof(command), commandsFile) != NULL){
169                         //Send a "false connection" message to the UDP server listening socket (ls_UDP)
170                 if (sendto (s, " ", 1,0, (struct sockaddr *)&servaddr_in_copia, addrlen) == -1) {
171                         perror(argv[0]);
172                         fprintf(stderr, "%s: unable to send request to \"connect\" \n", argv[0]);
173                         exit(1);
174                 }
175
176                 //Waits for the response of the server with the new socket it has to talk to
177                 if(-1 == recvUDP(s, tmp, 1, &servaddr_in, &addrlen)){
178                         exit(1);
179                 }
180
```

Al iniciarse el bucle while cuando el cliente lee los comandos de los ficheros de órdenes, lo primero que hace es establecer la conexión con el socket de escucha, por eso observamos en el sendTo que se pasa la estructura &servaddr_in_copia, que ha sido definida anteriormente que es por la cual se define la dirección del servidor.

Posteriormente ocurre el recvUDP, es decir el socket de escucha ha creado un nuevo socket por el cual se va a comunicar el cliente. Ésta información se guarda en &servaddr_in, la cual se utiliza para el envío del comando por

parte del cliente y los recvUDP de los próximos mensajes que conteste el servidor.

Se observa que cada vez que se obtiene un comando del fichero ordenes*.txt se establece la conexión de nuevo con el socket de escucha y este le responde al cliente creando un nuevo socket para que se efectúe la comunicación.

El resto de cosas es igual a TCP.

PRUEBAS

Si lanzamos el servidor con las siguientes órdenes en **TCP**:

```
../bin/server
../bin/client localhost TCP ordenes1.txt &
```

ordenes1.txt:

```
DAME /index.html k
GET /index.html k
GET /docencia.html k
GET /ejemplo1.html k
GET /ejemplo2.html k
GET /ejemplo3.html k
GET /ejemplo4.html c
```

Se crearán dos archivos, el del cliente con el nombre de su puerto efímero y el http.log.

El archivo del Cliente contendrá el comando tomado del archivo, envíado como petición al Servidor y las 6 respuestas del Servidor.

Comando: **DAME /index.html k**

```
[TCP] Connected to localhost on port 55174 at Tue Dec 14 00:32:46 2021

C: DAME /index.html k
S: HTTP/1.1 501 Not Implemented
S: Server: Servidor de Javier
S: Connection: keep-alive
S: Content-Length: 60
S:
S: <html><body><h1> 501 Not Implemented </h1></body></html>
```

El archivo del http.log:

```
FECHA: 14-12-2021 | HORA: 00:32:46 | NOMBRE HOST: localhost | IP: 127.0.0.1 | PROTOCOLO: TCP | PUERTO CLIENTE: 55174
FECHA: 14-12-2021 | COMANDO: (DAME /index.html k)
FECHA: 14-12-2021 | RESPUESTA: (HTTP/1.1 501 Not Implemented
)
FECHA: 14-12-2021 | RESPUESTA: (Server: Servidor de Javier
)
FECHA: 14-12-2021 | RESPUESTA: (Connection: keep-alive
)
FECHA: 14-12-2021 | RESPUESTA: (Content-Length: 60)
FECHA: 14-12-2021 | RESPUESTA: (
)
FECHA: 14-12-2021 | RESPUESTA: (<html><body><h1> 501 Not Implemented </h1></body></html>)
```

Comando: **GET /index.html k**

```
C: GET /index.html k
S: HTTP/1.1 200 OK
S: Server: Servidor de Javier
S: Connection: keep-alive
S: Content-Length: 98850
S:
S: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1
xmlns:dc="http://purl.org/dc/terms/"  xmlns:foaf="
xmlns:sioct="http://rdfs.org/sioc/types#"  xmlns:s
<meta charset="utf-8">    <meta name="viewport" co
home-matricula-2021" /><link rel="shortlink" href="
Universidad de Salamanca</title>      <link href="
OpenSans.css' rel='stylesheet' type='text/css'>
qypbsz");@import url("https://www.usal.es/modules/
modules/system/system.theme.css?qypbsz");</style><
www.usal.es/sites/all/modules/contrib/date/date_po
modules/node/node.css?qypbsz");@import url("https:
```

El archivo del http.log:

```
FECHA: 14-12-2021 | COMANDO: (GET /index.html k)
FECHA: 14-12-2021 | RESPUESTA: (HTTP/1.1 200 OK
)
FECHA: 14-12-2021 | RESPUESTA: (Server: Servidor de Javier
)
FECHA: 14-12-2021 | RESPUESTA: (Connection: keep-alive
)
FECHA: 14-12-2021 | RESPUESTA: (Content-Length: 98850)
FECHA: 14-12-2021 | RESPUESTA: (
850)
FECHA: 14-12-2021 | RESPUESTA: (<!DOCTYPE html PUBLIC "-//W3C//DTD
1.0/modules/content/"  xmlns:dc="http://purl.org/dc/terms/"  xmlns:
xmlns:sioc="http://rdfs.org/sioc/ns#"  xmlns:sioct="http://rdfs.org/
profile="http://www.w3.org/1999/xhtml/vocab">    <meta charset="utf
html; charset=utf-8" /><link rel="canonical" href="/home-matricula-
type="image/png" />    <title>Universidad de Salamanca | Universida
href='/sites/all/themes/momentum/css/fonts/OpenSans.css' rel='style
www.usal.es/modules/system/system.base.css?qypbsz");@import url("ht
qypbsz");@import url("https://www.usal.es/modules/system/system.the
date.css?qypbsz");@import url("https://www.usal.es/sites/all/module
qypbsz");@import url("https://www.usal.es/modules/node/node.css?qyp
qypbsz");@import url("https://www.usal.es/sites/all/modules/contrib
style><style type="text/css" media="all">@import url("https://www.u
flexslider_format/css/flexslider.css?qypbsz");@import url("https://
sites/all/modules/contrib/glazed_builder/glazed_builder/css/social.
qypbsz");@import url("https://www.usal.es/sites/all/modules/contrib
```

Comando: **GET /ejemplo1.html k**

```
C: GET /ejemplo1.html k
S: HTTP/1.1 404 Not Found
S: Server: Servidor de Javier
S: Connection: keep-alive
S: Content-Length: 60
S:
S: <html><body><h1> 404 Not Implemented </h1></body></html>
```

El archivo del http.log:

```
FECHA: 14-12-2021 | COMANDO: (GET /ejemplo1.html k)
FECHA: 14-12-2021 | RESPUESTA: (HTTP/1.1 404 Not Found
)
FECHA: 14-12-2021 | RESPUESTA: (Server: Servidor de Javier
)
FECHA: 14-12-2021 | RESPUESTA: (Connection: keep-alive
)
FECHA: 14-12-2021 | RESPUESTA: (Content-Length: 60)
FECHA: 14-12-2021 | RESPUESTA: (
)
FECHA: 14-12-2021 | RESPUESTA: (<html><body><h1> 404 Not Implemented </h1></body></html>)
```

ordenes3.txt:

```
GET /index.html k
GET /docencia.html k
GET /ejemplo.html c
DAME /index.htm c
```

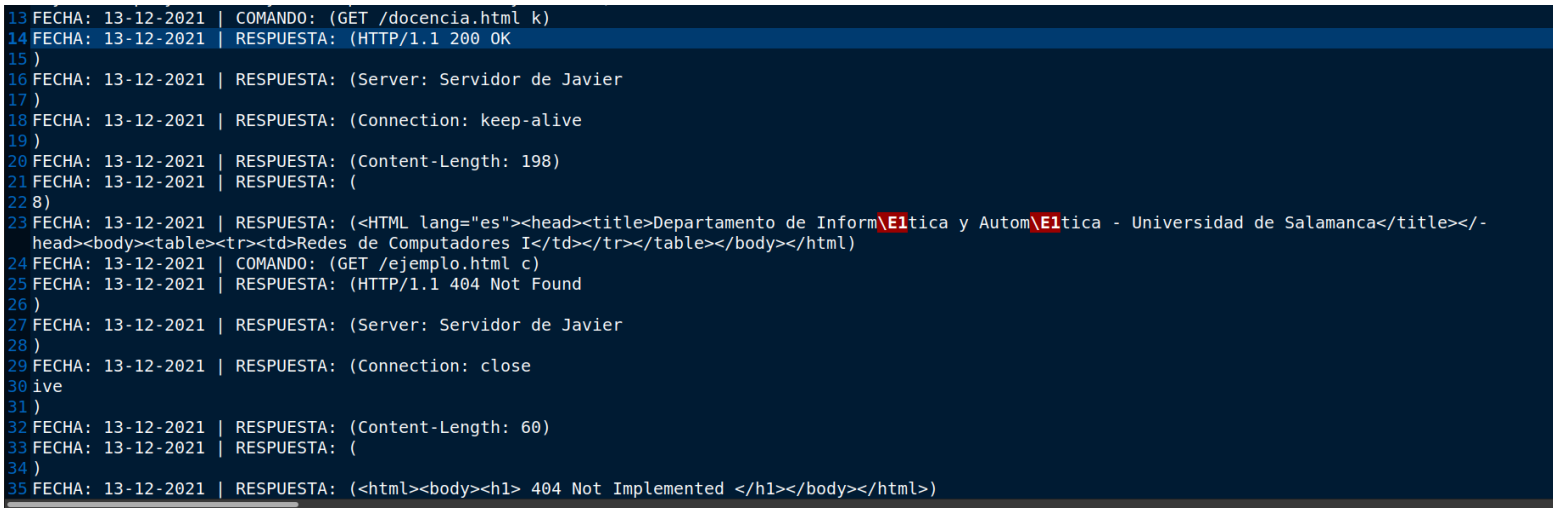
Observamos el resultado por parte del cliente:

```
[TCP] Connected to localhost on port 56726 at Tue Dec 13 04:48:33 2021

C: GET /index.html k
S: HTTP/1.1 200 OK
S: Server: Servidor de Javier
S: Connection: keep-alive
S: Content-Length: 98850
S:
S: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd"><html lang="es" dir="ltr"
xmlns:content="http://purl.org/rss/1.0/modules/content/"  xmlns:dc="http://purl.org/dc/terms/"  xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:og="http://ogp.me/ns#"  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  xmlns:sioc="http://rdfs.org/sioc/ns#"  xmlns:sioct="http://rdfs.org/-
sioc/types#"  xmlns:skos="http://www.w3.org/2004/02/skos/core#"  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">  <head profile="http://www.w3.org/1999/
xhtml/vocab">    <meta charset="utf-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <meta http-equiv="Content-Type"
content="text/html; charset=utf-8" /><link rel="canonical" href="/home-matricula-2021" /><link rel="shortlink" href="/node/103040" /><link
rel="shortcut icon" href="https://www.usal.es/files/favicon.png" type="image/png" />    <title>Universidad de Salamanca | Universidad de Salamanca</-
title>  <link href='/sites/all/themes/momentum/css/fonts/raleway.css' rel='stylesheet' type='text/css'> <link href='/sites/all/themes/momentum/css/-
fonts/OpenSans.css' rel='stylesheet' type='text/css'>   <!--  Ejecutado desde: Desconocido        -->    <style type="text/css" media="all">@import
url("https://www.usal.es/modules/system/system.base.css?qypbsz");@import url("https://www.usal.es/modules/system/system.menus.css?qypbsz");@import
url("https://www.usal.es/modules/system/system.messages.css?qypbsz");@import url("https://www.usal.es/modules/system/system.theme.css?qypbsz");</-
style><style type="text/css" media="all">@import url("https://www.usal.es/sites/all/modules/contrib/date/date_api/date.css?qypbsz");@import
url("https://www.usal.es/sites/all/modules/contrib/date/date_popup/themes/datepicker.1.7.css?qypbsz");@import url("https://www.usal.es/modules/field/-
theme/field.css?qypbsz");@import url("https://www.usal.es/modules/node/node.css?qypbsz");@import url("https://www.usal.es/modul../orders/ordenes3.txt
C: GET /docencia.html k
S: HTTP/1.1 200 OK
S: Server: Servidor de Javier
S: Connection: keep-alive
S: Content-Length: 198
S:
S: <HTML lang="es"><head><title>Departamento de Informática y Automática - Universidad de Salamanca</title></head><body><table><tr><td>Redes de
Computadores I</td></tr></table></body></html
C: GET /ejemplo.html c
S: HTTP/1.1 404 Not Found
S: Server: Servidor de Javier
S: Connection: close
S: Content-Length: 60
S:
S: <html><body><h1> 404 Not Implemented </h1></body></html>
C: DAME /index.htm c
```

Podemos observar que los tres primeros comandos son enviados correctamente por parte del cliente ya que observamos una respuesta correcta por parte del servidor. Sin embargo nos fijamos en que el último comando no recibe respuestas por parte del servidor. Esto es debido a que el comando anterior **GET /ejemplo.html c** finaliza la conexión. Al ser **TCP** dicha letra en el comando corta la conexión con el servidor por lo que el siguiente comando que envía el cliente no obtiene respuesta.

En cuanto al http.log que guarda la información que recibe y envía el servidor observamos el último comando que recibe es **GET /ejemplo.html c**:



Si lanzamos el servidor con las siguientes órdenes en **UDP**:

```
../bin/client localhost UDP ordenes1.txt &
../bin/client localhost UDP ordenes2.txt &
../bin/client localhost UDP ordenes3.txt &
```

Vamos a ver ordenes3.txt:

```
GET /index.html k
GET /docencia.html k
GET /ejemplo.html c
DAME /index.htm c
```

Comando: **GET /index.html k**

```
[UDP] "False connected" to localhost on port 52042 at Tue Dec 14 01:00:57 2021


C: GET /index.html k
S: HTTP/1.1 200 OK
S: Server: Servidor de Ã lvaro
S: Connection: close
S: Content-Length: 98850
S:
S: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"  "http://www.w3.org/MarkUp/DTD/xhtml-
xmlns:dc="http://purl.org/dc/terms/"  xmlns:foaf="http://xmlns.com/foaf/0.1/"  xmlns:og="http:/
xmlns:sioct="http://rdfs.org/sioc/types#"  xmlns:skos="http://www.w3.org/2004/02/skos/core#"  x
<meta charset="utf-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0"
home-matricula-2021" /><link rel="shortlink" href="/node/103040" /><link rel="shortcut icon" hr
Universidad de Salamanca</title>      <link href='/sites/all/themes/momentum/css/fonts/raleway.
OpenSans.css' rel='stylesheet' type='text/css'>       <!--  Ejecutado desde: Desconocido      -
qypbsz");@import url("https://www.usal.es/modules/system/system.menus.css?qypbsz");@import url(
modules/system/system.theme.css?qypbsz");</style><style type="text/css" media="all">@import url
www.usal.es/sites/all/modules/contrib/date/date_popup/themes/datepicker.1.7.css?qypbsz");@impor
modules/node/node.css?qypbsz");@import url("https://www.usal.es/modul
```

El archivo del http.log:

```
FECHA: 14-12-2021 | HORA: 01:00:57 | NOMBRE HOST: localhost | IP: 127.0.0.1 | PROTOCOLO: UDP | PUERTO CLIENTE: 52042
FECHA: 14-12-2021 | HORA: 01:00:57 | NOMBRE HOST: localhost | IP: 127.0.0.1 | PROTOCOLO: UDP | PUERTO CLIENTE: 55483
FECHA: 14-12-2021 | RESPUESTA: (HTTP/1.1 200 OK
)
FECHA: 14-12-2021 | RESPUESTA: (Server: Servidor de Álvaro
)
FECHA: 14-12-2021 | RESPUESTA: (Connection: close
)
FECHA: 14-12-2021 | RESPUESTA: (Content-Length: 98850)
FECHA: 14-12-2021 | RESPUESTA: (
850)
FECHA: 14-12-2021 | RESPUESTA: (<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"  "http://www.w3.org/MarkUp/DTD/
1.0/modules/content/"  xmlns:dc="http://purl.org/dc/terms/"  xmlns:foaf="http://xmlns.com/foaf/0.1/"  xmlns:og="http:/
xmlns:sioc="http://rdfs.org/sioc/ns#"  xmlns:sioct="http://rdfs.org/sioc/types#"  xmlns:skos="http://www.w3.org/2004/0
profile="http://www.w3.org/1999/xhtml/vocab">    <meta charset="utf-8">    <meta name="viewport" content="width=device
html; charset=utf-8" /><link rel="canonical" href="/home-matricula-2021" /><link rel="shortlink" href="/node/103040" /
type="image/png" />    <title>Universidad de Salamanca | Universidad de Salamanca</title>    <link href='/sites/all/th
href='/sites/all/themes/momentum/css/fonts/OpenSans.css' rel='stylesheet' type='text/css'>  <!--  Ejecutado desde: De
www.usal.es/modules/system/system.base.css?qypbsz");@import url("https://www.usal.es/modules/system/system.menus.css?q
qypbsz");@import url("https://www.usal.es/modules/system/system.theme.css?qypbsz");</style><style type="text/css" medi
date.css?qypbsz");@import url("https://www.usal.es/sites/all/modules/contrib/date/date_popup/themes/datepicker.1.7.css
qypbsz");@import url("https://www.usal.es/modules/node/node.css?qypbsz");@import url("https://www.usal.es/modules/sear
qypbsz");@import url("https://www.usal.es/sites/all/modules/contrib/views/css/views.css?qypbsz");@import url("https://
```

Comando: **GET /ejemplo.html c**

```
21 [UDP] "False connected" to localhost on port 52042 at Tue Dec 14 01:00:57 2021
22
23
24 C: GET /ejemplo.html c
25 S: HTTP/1.1 404 Not Found
26 S: Server: Servidor de Ã lvaro
27 S: Connection: close
28 S: Content-Length: 60
29 S:
30 S: <html><body><h1> 404 Not Implemented </h1></body></html>
```

El archivo del http.log:

```
193 FECHA: 14-12-2021 | HORA: 01:00:58 | NOMBRE HOST: localhost | IP: 127.0.0.1 | PROTOCOLO: UDP | PUERTO CLIENTE: 52042
194 FECHA: 14-12-2021 | RESPUESTA: (HTTP/1.1 404 Not Found
195 )
196 FECHA: 14-12-2021 | RESPUESTA: (Server: Servidor de Álvaro
197 )
198 FECHA: 14-12-2021 | RESPUESTA: (Connection: close
199 )
200 FECHA: 14-12-2021 | RESPUESTA: (Content-Length: 60)
201 FECHA: 14-12-2021 | RESPUESTA: (
202 )
203 FECHA: 14-12-2021 | RESPUESTA: (<html><body><h1> 404 Not Implemented </h1></body></html>)
```

Comando: **DAME /index.htm c**

```
34 C: DAME /index.htm c
35 S: HTTP/1.1 501 Not Implemented
36 S: Server: Servidor de Ã lvaro
37 S: Connection: close
38 S: Content-Length: 60
39 S:
40 S: <html><body><h1> 501 Not Implemented </h1></body></html>
41
42 [UDP] All done at Tue Dec 14 01:00:58 2021
```

El archivo del http.log:

```
204 FECHA: 14-12-2021 | RESPUESTA: (HTTP/1.1 501 Not Implemented
205 )
206 FECHA: 14-12-2021 | RESPUESTA: (Server: Servidor de Álvaro
207 )
208 FECHA: 14-12-2021 | RESPUESTA: (Connection: close
209 )
210 FECHA: 14-12-2021 | HORA: 01:00:58 | NOMBRE HOST: localhost | IP: 127.0.0.1 | PROTOCOLO:
211 FECHA: 14-12-2021 | RESPUESTA: (Content-Length: 60)
212 FECHA: 14-12-2021 | RESPUESTA: (
213 )
214 FECHA: 14-12-2021 | RESPUESTA: (<html><body><h1> 501 Not Implemented </h1></body></html>)
```

Observamos como en el caso de **UDP** sí se llega a recibir este comando por parte del servidor, debido a que por **UDP** establecemos la conexión cada vez que enviamos un comando nuevo.Esto es porque enviamos el comando al socket de escucha y este crea

un socket por el cual nos comunicaremos. Cada vez que se envía un comando crea uno nuevo.

README

Para hacer make simplemente ir a la carpeta src, abrir una terminal y escribir make. Dicho comando compila todo. Para ejecutar el programa ir a la carpeta orders y desde ahí ejecutar **./lanzarServidor.sh**

Para poder observar los logs, simplemente debe dirigirse a la carpeta bin, dentro de ella se observará la carpeta logs. Ahí es donde se generarán.

Para eliminar el servidor y el cliente en la carpeta src donde se encuentra el make, ejecutar en un terminal el comando **make stop**. Para matar al cliente basta con **pkill client**.

CONCLUSIONES

La práctica tanto en local como en nogal cumple parcialmente con los requisitos, pero no hemos conseguido resolver el problema de index.html debido a su gran cantidad de bytes. Funciona pero trunca la string con el contenido del fichero index.html que es enviado por el servidor al cliente.

Sin embargo, cualquier otro tipo de comando funciona correctamente tanto en nogal como en local.