VNiVERSiDAD
D SALAMANCA

Práctica Sockets: Servicio Finger

Redes de Computadores I

Grado en Ingeniería Informática

Diego Borrallo Herrero 49367527M
Jaime Castellanos Rubio 71047797S

ÍNDICE

- ## Introducción:
  Para la realización de la práctica se han utilizado sockets de Berkeley implementados en C y un sistema operativo Linux Debian.
  Su objetivo es implementar un cliente similar a un “finger” y cuyo servidor al que se accede devuelva la información formateada como este servicio dependiendo de los parámetros pasados al cliente en su ejecución.

- ## Aspectos relevantes:
  - **Planificación:**
    Para el formato de las respuestas, primero se detectaron las variables que serían esenciales en el programa, es decir, cómo se devolvía la información de un servicio finger según qué casos.
    Por ello, se planificó de la siguiente manera:

```
// Plan para obtener info específica del usuario(si existe el usuario):
Login desde getent
Name desde getent
Directory desde getent
Shell desde getent
TTY desde lastlog
IP desde lastlog
Tiempo de conexion desde lastlog
Mail suele estar vacío
Plan suele estar vacío

// Plan para obtener info de todos los usuarios
Login desde who
Nombre desde getent (ir haciendo uno a uno)
TTY desde lastlog
Idle desde w
Login time desde lastlog
Office(IP) desde lastlog
Office Phone suele estar vacío
```

  - **Obtención de la información:**
    Una vez se sabía qué información iba a ser necesaria y desde dónde se podía obtener, pasamos a implementar el proceso de obtención de la información en C, ejecutando los comandos mediante “popen”, que se encarga de ejecutar un proceso con el comando especificado y almacenar la salida de este en un fichero del que se puede sacar la información tratando la salida de distintas maneras.

- **Comunicación entre procesos:**
  Para la parte de comunicación entre procesos, la arquitectura a seguir era cliente-servidor tanto para TCP como para UDP.
  Por ello, en TCP se crea un proceso hijo mediante un socket de escucha para cada cliente que desee conectarse y en UDP se ha implementado de la misma manera, crearemos un proceso hijo para cada cliente que se conecte por el socket de escucha del proceso demonio.
  A cada proceso se le asigna un puerto efímero para que no se interfiera en la comunicación con el servidor.
  Para la creación del proceso hijo que atienda clientes UDP se sigue la siguiente estructura:
    - El cliente manda un primer mensaje al socket de escucha para solicitar conexión.
    - Cuando el servidor recibe el mensaje crea un nuevo socket con un puerto efímero aleatorio.
    - El servidor manda a ese cliente por el mismo socket el número de puerto asignado.
    - Cuando el cliente recibe el número de puerto, lo actualiza y ya se comunica con el servidor por este puerto nuevo.

- **Logs de mensajes:**
  Se pedía crear un fichero para cada cliente con el nombre del puerto efímero asignado a cada cliente. Pero esto presentaba un problema, el acceso concurrente al fichero donde se quiere guardar la información.
  Por ello hemos implementado un semáforo que afecta a cada cliente para solucionar problemas de “Violación de segmento”, es decir, accesos a memoria no permitidos.
  Básicamente se ha creado una función llamada “*escribe_log*” que recibe el mensaje a escribir y el número de puerto efímero, por lo que dentro de esta función se maneja toda la lógica del semáforo, tanto abrirlo como cerrarlo y el acceso al fichero.
  Finalmente cada proceso libera los recursos adquiridos, en este caso el semáforo.

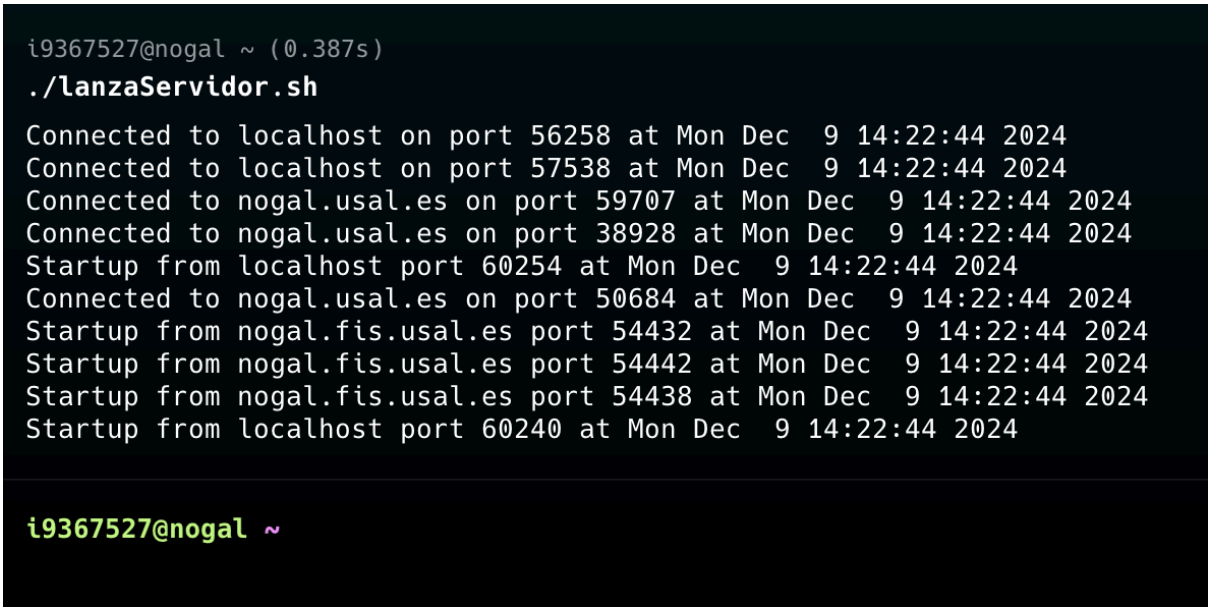
- **Timeout del servidor:**
  Se ha implementado una manejadora de SIGALRM en el proceso demonio para que, pasado un cierto tiempo(configurado a 5 minutos por nosotros), este proceso finalice la escucha, es decir, no se quede ejecutando por siempre.

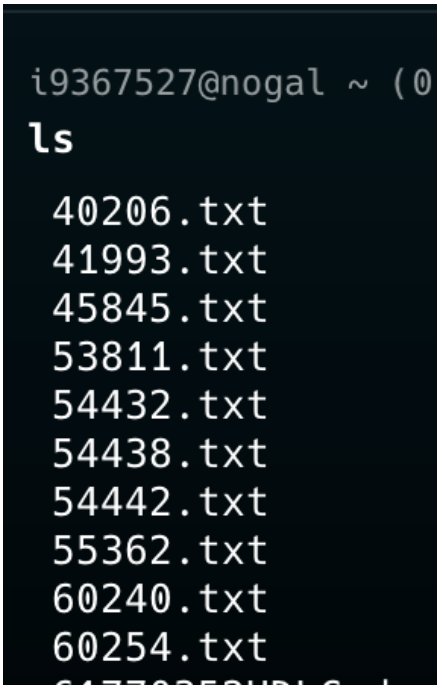
- **Documentación de pruebas realizadas:**

  Para las pruebas realizadas primero se han ido lanzando poco a poco los diferentes clientes con los diferentes casos de parámetros que se pueden dar, es decir:

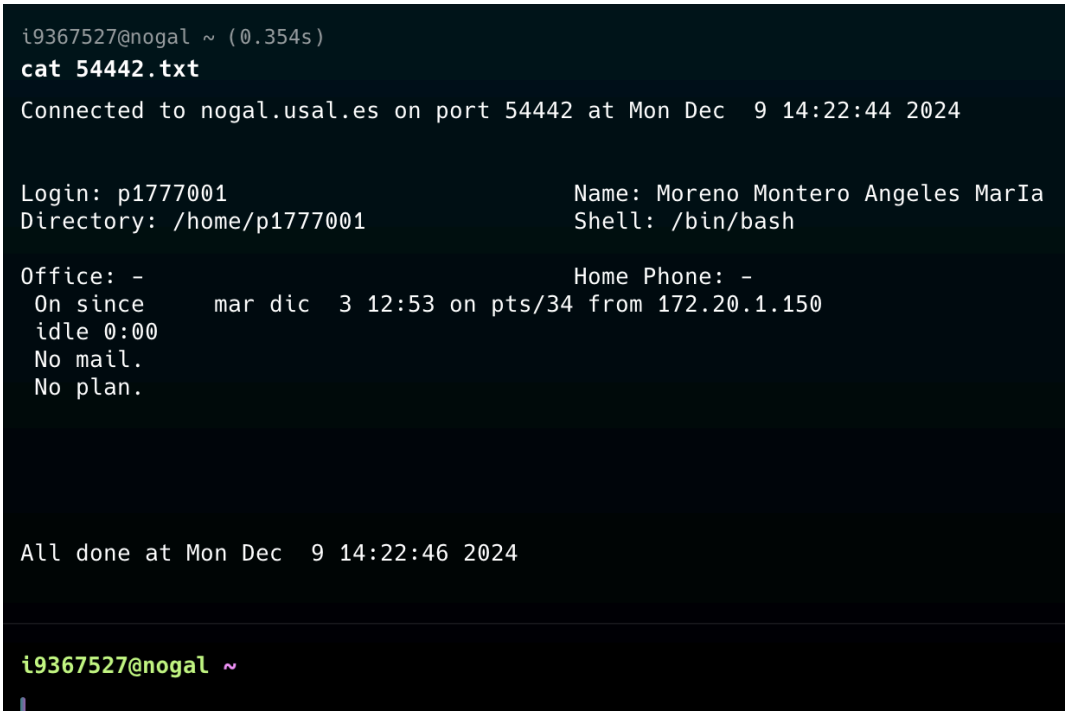
  - Argumento vacío -> finger -l
  - usuario -> finger usuario
  - @host-> finger -l con el servidor en el host
  - usuario@host -> finger usuario con el servidor en el host

  Una vez hechas y depuradas estas pruebas se empezó a probar el script "*lanzaServidor.sh*", con el cual se deben crear diez archivos cada uno con el nombre del puerto efímero de cada cliente, como se puede ver en la siguiente captura:

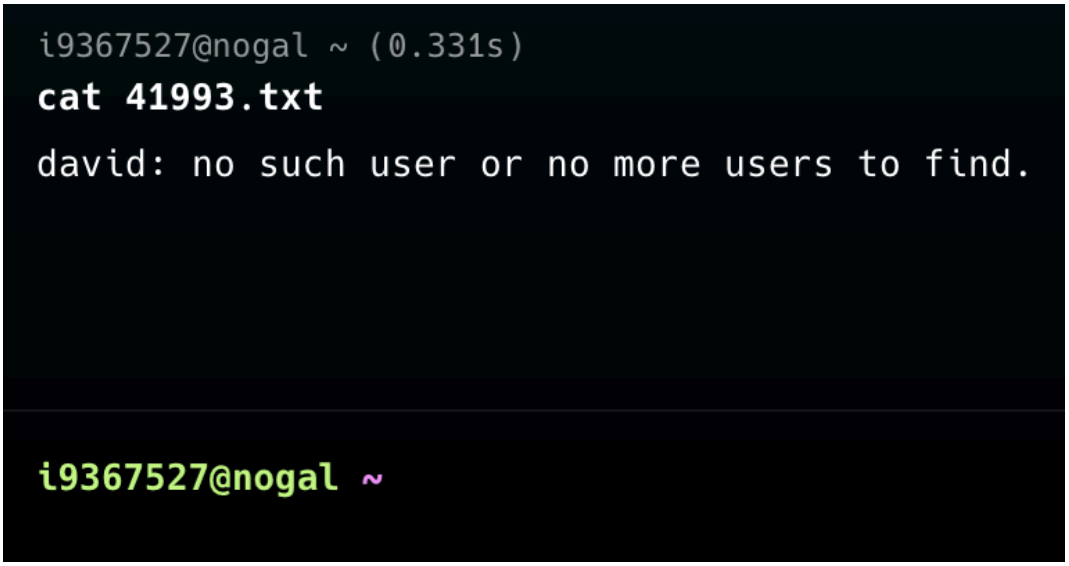




Si obtenemos la información de cada archivo con el comando “**cat nombre_archivo.txt**” podremos ver el correcto funcionamiento de los comandos, como por ejemplo el comando “**./cliente TCP p1777001@nogal.usal.es**” nos da la siguiente salida, que corresponde a la información del usuario “p1777001” que se encuentra en el host “nogal.usal.es”:



El comando “**./cliente UDP david**” nos da la siguiente salida, que indica que no se encuentra ningún “david” con minúscula almacenado en el servidor:



El comando “**./cliente TCP @nogal.usal.es**” nos da la siguiente salida, que corresponde con todos los usuarios que están activos en este momento en ell host “nogal.usal.es”:

```
i9367527@nogal ~ (0.386s)
cat 54432.txt
Connected to nogal.usal.es on port 54432 at Mon Dec  9 14:22:44 2024


Login: i1120512                         Name: Fuentes Garcia Cynthia
Directory: /home/i1120512               Shell: /bin/bash

Office: -                               Home Phone: -
 On since    lun dic  9 13:53 on pts/53 from 85.48.188.109
 idle 1:52m
 No mail.
 No plan.



Login: i1120512                         Name: Fuentes Garcia Cynthia
Directory: /home/i1120512               Shell: /bin/bash

Office: -                               Home Phone: -
 On since    lun dic  9 13:53 on pts/53 from 85.48.188.109
 idle 1:52m
 No mail.
 No plan.



Login: i0919688                         Name: Calzada Bernal Juan
Directory: /home/i0919688               Shell: /bin/bash

Office: -                               Home Phone: -
 On since      lun dic  9 13:41 on pts/41 from 10.50.17.33
 idle 8:08
 No mail.
 No plan.

i9367527@nogal ~
```

El comando “**./cliente TCP**” nos da la siguiente salida, que se corresponde con todos los usuarios que se encuentran activos en este momento en el equipo local donde se encuentre el servidor.

```
i9367527@nogal ~ (0.37s)
cat 60254.txt
Connected to localhost on port 60254 at Mon Dec  9 14:22:44 2024


Login: i1120512                         Name: Fuentes Garcia Cynthia
Directory: /home/i1120512               Shell: /bin/bash

Office: -                               Home Phone: -
 On since    lun dic  9 13:53 on pts/53 from 85.48.188.109
 idle 1:52m
 No mail.
 No plan.



Login: i1120512                         Name: Fuentes Garcia Cynthia
Directory: /home/i1120512               Shell: /bin/bash

Office: -                               Home Phone: -
 On since    lun dic  9 13:53 on pts/53 from 85.48.188.109
 idle 1:52m
 No mail.
 No plan.



Login: i0919688                         Name: Calzada Bernal Juan
Directory: /home/i0919688               Shell: /bin/bash

Office: -                               Home Phone: -
 On since      lun dic  9 13:41 on pts/41 from 10.50.17.33
 idle 8:08
 No mail.
 No plan.

i9367527@nogal ~
```