Implementación sobre Máquina Virtual de un servicio en KALI LINUX

Luis Alfredo López, lualopezpe@unal.edu.co
Yeison Ortiz Cano, yeortizca@unal.edu.co

I. Introducción

En el desarrollo del curso de Sistemas Operativos se examinó la gestión de procesos en distintos S.O, aspecto que es muy importante en el diseño de los mismo. Ahora bien, como parte fundamental del curso, se presenta a continuación la implementación de un proceso Daemon (servicio) en un S.O Linux. El programa que se presenta está programado en lenguaje C y corre sobre una Máquina Virtual. El servicio que se mostrará consiste en un *Servidor Echo* que recibe mensaje de múltiples *Clientes.*

II. Proceso Daemon

A. *Definición*

Un proceso Daemon es un proceso que corre en segundo plano y no tiene terminal de control. Esto es, requiere de muy poca interacción con el usuario. Los procesos Daemon se diseñan con el fin de proveer servicios. Ejemplos muy comunes de procesos Daemon son aquellos que están encargados de observar la actividad en la red.

B. *Diseño de un proceso Daemon*

Un proceso Daemon se diseña como cualquier otro proceso, a diferencia de algunos aspectos fundamentales. Una pequeña guía para la creación de un Daemon sería la siguiente:

1. Crear un proceso normal (Padre).
2. Crear un proceso Hijo, del padre anterior.
3. Terminar el proceso Padre. El proceso Hijo queda huérfano y es tomado por el proceso Init.
4. Hacer la llamada a la función setsid() para correr el proceso en una nueva sesión y tener un nuevo grupo.
5. Después del paso anterior, se puede decir que tenemos un proceso Daemon que no pertenece a ninguna terminal.
6. Se cambia el directorio de trabajo del proceso Daemon a *"/root"* y se cierran los descriptores de archivos stdin, stdout, stderr.
7. Se incorpora el programa principal del Daemon.

El siguiente código de nuestro Daemon recoge los pasos del 1 al 6.

```
    /* PASO 1.*/
static void skeleton_daemon()
{
    pid_t pid;
    /* PASO 2. Fork off the parent process */
    pid = fork();

    /* An error occurred */
    if (pid < 0)
        exit(EXIT_FAILURE);

    /* PASO 3. Success: Let the parent
terminate */
    if (pid > 0)
        exit(EXIT_SUCCESS);

    /* PAS0 4. On success: The child process
becomes session leader */
    if (setsid() < 0)
        exit(EXIT_FAILURE);

    /* Catch, ignore and handle signals */
    //TODO: Implement a working signal handler
*/
    signal(SIGCHLD, SIG_IGN);
    signal(SIGHUP, SIG_IGN);

    /* Fork off for the second time*/
    pid = fork();

    /* An error occurred */
    if (pid < 0)
        exit(EXIT_FAILURE);

    /* PASO 5. Success: Let the parent
terminate */
    if (pid > 0)
        exit(EXIT_SUCCESS);

    /* Set new file permissions */
    umask(0);
```

```
    /* PASO 6. Change the working directory to
the root directory */
    /* or another appropriated directory */
    chdir("/");

    /* Close all open file descriptors */
    int x;
    for (x = sysconf(_SC_OPEN_MAX); x>=0; x--)
    {
        close (x);
    }

    /* Open the log file */
    openlog ("firstdaemon", LOG_PID,
LOG_DAEMON);
}
```

Con la función anterior podemos 'Demonizar' nuestro proceso.

C. Programa Principal

Nuestro programa principal consiste de un 'Servidor Echo', el cual recibe un mensaje de un 'Cliente' (Umesh Toke, 2015). La intercomunicación entre estos procesos se ha a través de sockets. Adicional, nuestro servidor es capaz de manejar varias conexiones al mismo tiempo utilizando hilos. Esto es, un 'Cliente' solicita conexión (evento) con el 'Servidor', esta conexión se establece a través de un hilo (thread) y en caso de que un nuevo 'Cliente' solicite conexión, a esta se le asigna un hilo distinto. El código del 'Servidor' y del 'Cliente' se adjuntan a este documento.

III. Máquina Virtual

Se utilizó VirtualBox para correr la máquina virtual, el S.O escogido fue KALI LINUX.

Figura 1. Kali Linux sobre Máquina Virtual de VBox.

Una vez instalado nuestro S.O sobre la M.V se necesitó de algunos pasos adicionales:

- ❏ Instalación de compilador GNU GCC.
- ❏ Instalación de editor de texto Gedit.
- ❏ Instalación de las Guest Additions para la comunicación entre el Host y la M.V a través de carpeta compartida y portapapeles.

IV. Metodología

- Se compilan server.c y client.c
  - gcc server.c -o server -pthread
  - gcc client.c -o client -pthread
- Se adiciona ./server al inicio de Kali Linux
  - Guardar server en /usr/sbin
    - cp server /usr/sbin/server
  - Crear script de inicio usando /etc/init.d/skeleton como referencia
  - Mover el script a /etc/init.d
    - sudo mv /etc/init.d/server
  - Otorgar al script permisos de ejecución
    - chmod 755 /etc/init.d/server
  - ir a /etc/init.d
    - cd /etc/init.d
  - Incluir nuestro script a la lista de inicio con baja prioridad
    - sudo update-rc.d server defaults 97 03
  - Reiniciar
  - Verificar que en efecto nuestro server esta corriendo como Daemon (PPID debe ser 1).
    - ps -xj | grep server
  - Ejecutar el cliente y esperar a que el servidor devuelva el mensaje.

V. Resultados

Se adjuntan algunos pantallazos que dan muestra de que en efecto nuestro servicio está implementado. En la implementación del proceso Daemon se logró lo siguiente:

- Instalación de distintos S.O en maquina virtual a través de la herramienta VirtualBox.
- Programación de un servicio en Linux a partir de una estructura básica de proceso Daemon.
- Utilización de sockets para la intercomunicación entre procesos.
- Manejo básico de hilos de procesos en C, thread.

- Adición de un servicio para ejecución en el startup de Linux.

VI. Conclusiones

- La implementación de un servicio en Linux es sencilla, aunque en algunas ocasiones puede resultar ser un tarea complicada.
- Las Máquinas Virtuales son un herramienta importante a la hora de hacer pruebas con Sistemas Operativos. Brindan mucha más seguridad que cuando se manipula un S.O de instalación limpia.
- La gestión de procesos es una de las tareas más importantes a realizar por el S.O.
- Siempre sobre un S.O hay una buena cantidad de procesos corriendo en segundo plano, encargados de la administración de servicios en nuestro sistema.

VII. Referencias

[1]UbuntuAsk, "How to correctly add a custom daemon to init.d?", *Askubuntu.com*, 2010. [Online]. Available: https://askubuntu.com/questions/18802/how-to-correctly-add-a-custom-daemon-to-init-d. [Accessed: 21- Apr- 2017].

[2]H. Arora, "Creating a Daemon Process in C Language with an Example Program", *Thegeekstuff.com*, 2017. [Online]. Available: http://www.thegeekstuff.com/2012/02/c-daemon-process. [Accessed: 21- Apr- 2017].

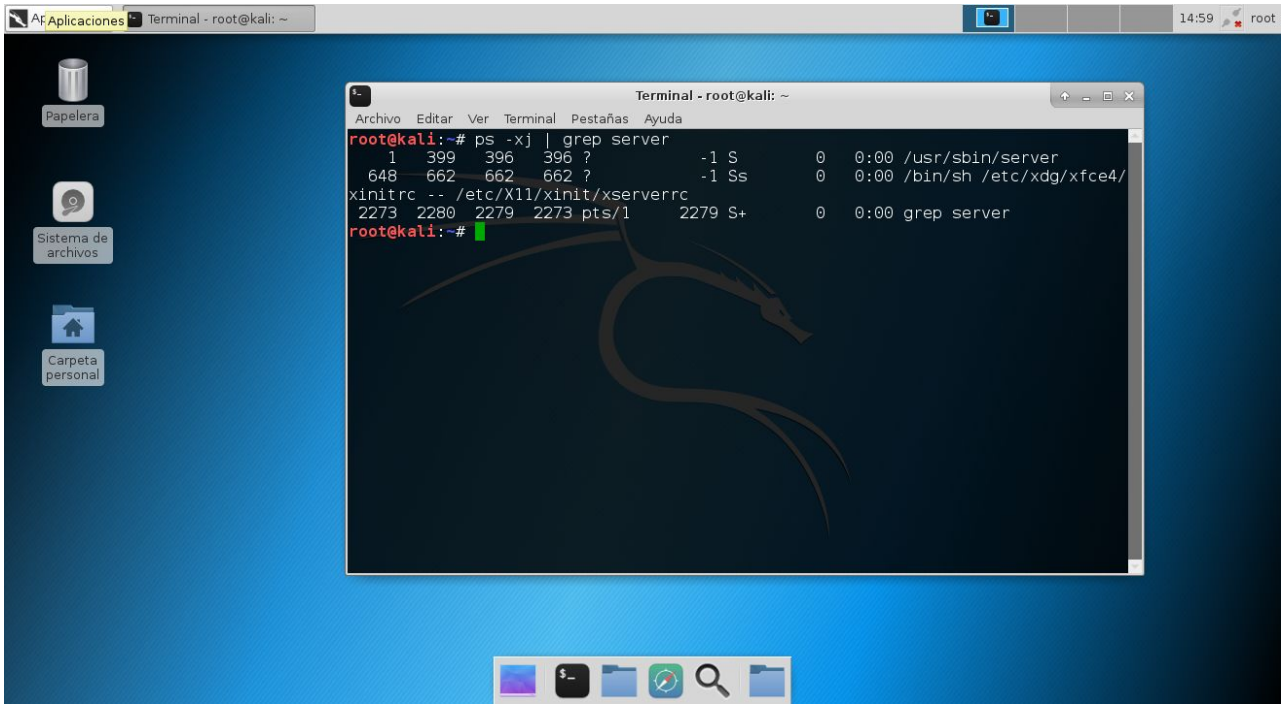
[3]D. Watson, "Linux Daemon Writing HOWTO", *Netzmafia.de*, 2017. [Online]. Available: http://www.netzmafia.de/skripten/unix/linux-daemon-howto.html. [Accessed: 21- Apr- 2017].

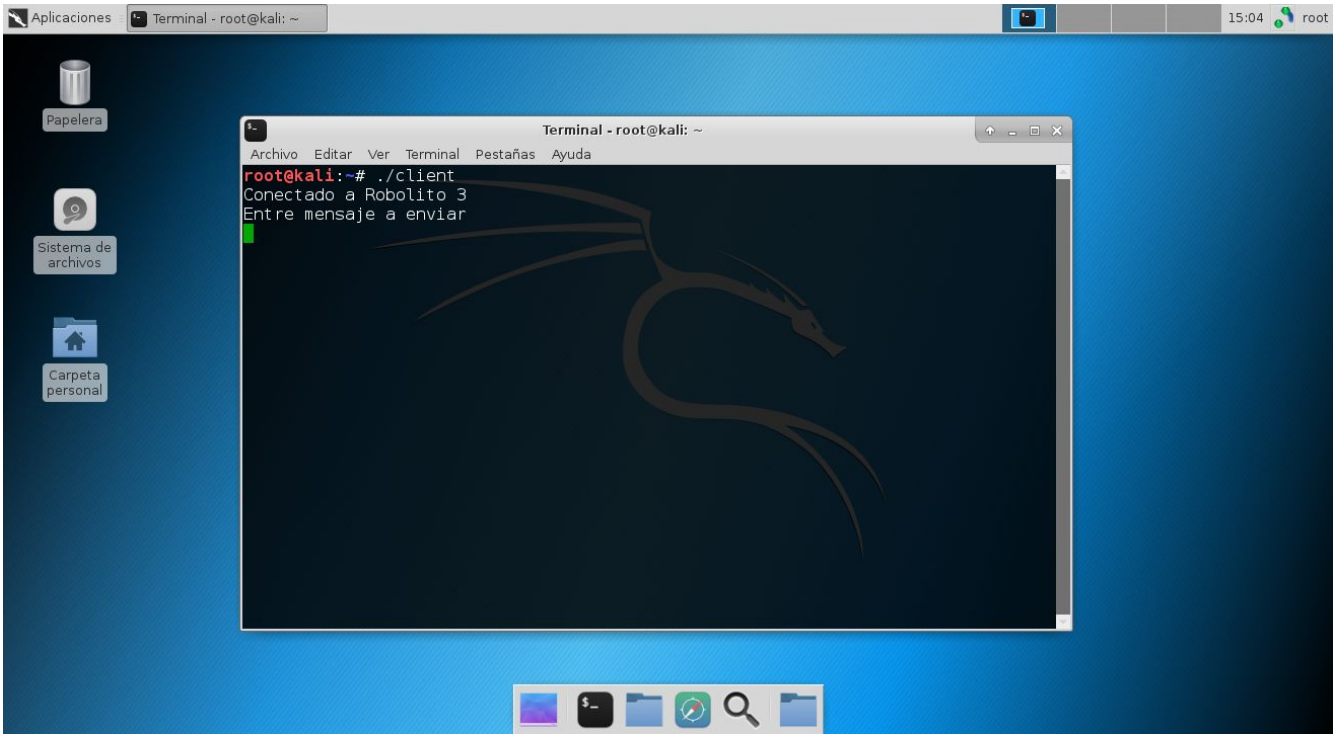
[4]B. Kurulamadi, "Unix Daemon Server Programming", *Enderunix.org*, 2001. [Online]. Available: http://www.enderunix.org/docs/eng/daemon.php. [Accessed: 21- Apr- 2017].

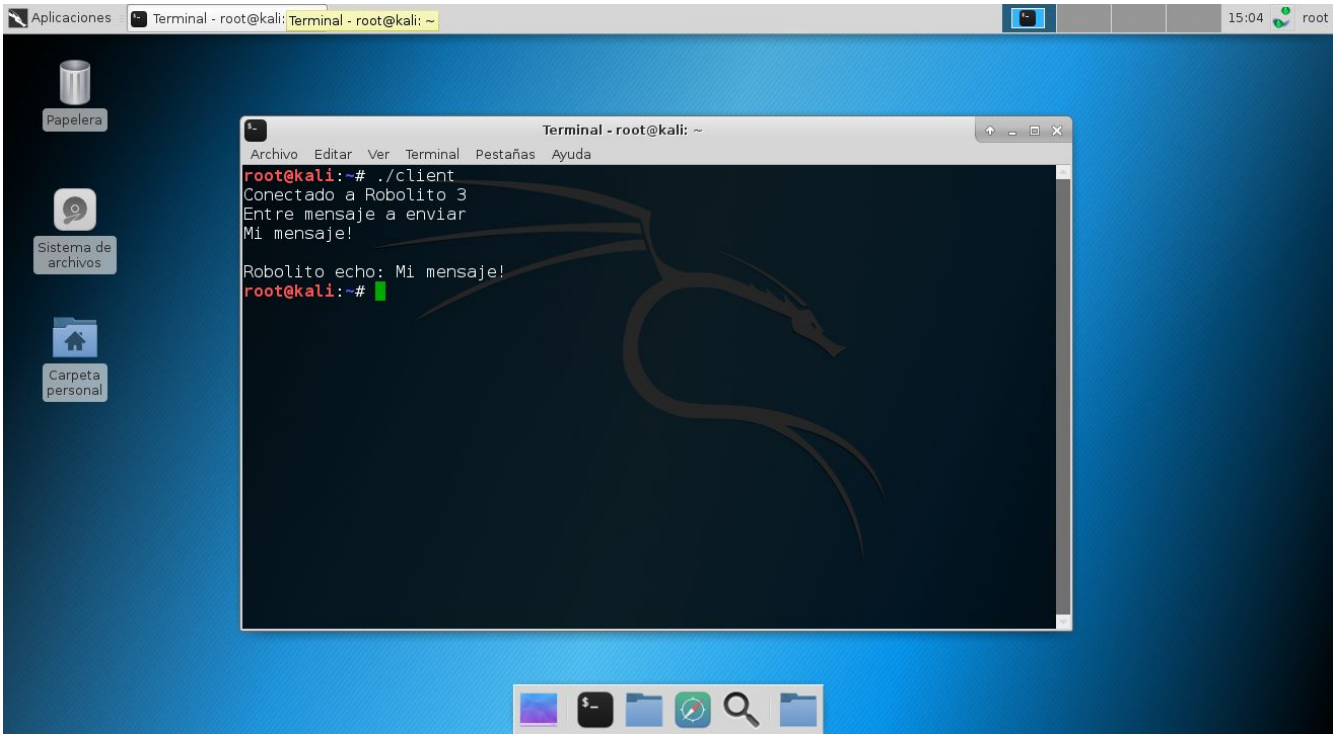
[5]howtos, "Linux Howtos: C/C++ -> Sockets Tutorial", *Linuxhowtos.org*, 2017. [Online]. Available: http://www.linuxhowtos.org/C_C++/socket.htm. [Accessed: 21- Apr- 2017].

[6]D. Bhaskar, "A simple chat program in C (TCP)", *Theinsanetechie.in*, 2017. [Online]. Available: http://www.theinsanetechie.in/2014/01/a-simple-chat-program-in-c-tcp.html. [Accessed: 21- Apr- 2017].

VIII. Anexos


Aplicaciones
Terminal - root@kali: ~
14:59
root
Papelera
Sistema de archivos
Carpeta personal
Terminal - root@kali: ~
Archivo Editar Ver Terminal Pestañas Ayuda
root@kali:~# ps -xj | grep server
1 399 396 396 ? -1 S 0 0:00 /usr/sbin/server
648 662 662 662 ? -1 Ss 0 0:00 /bin/sh /etc/xdg/xfce4/xinitrc -- /etc/X11/xinit/xserverrc
2273 2280 2279 2273 pts/1 2279 S+ 0 0:00 grep server
root@kali:~#


Aplicaciones
Terminal - root@kali: ~
15:04
root
Papelera
Sistema de archivos
Carpeta personal
Terminal - root@kali: ~
Archivo Editar Ver Terminal Pestañas Ayuda
root@kali:~# ./client
Conectado a Robolito 3
Entre mensaje a enviar


Aplicaciones
Terminal - root@kali:
Terminal - root@kali: ~
15:04
root
Papelera
Sistema de archivos
Carpeta personal
Terminal - root@kali: ~
Archivo Editar Ver Terminal Pestañas Ayuda
root@kali:~# ./client
Conectado a Robolito 3
Entre mensaje a enviar
Mi mensaje!
Robolito echo: Mi mensaje!
root@kali:~#