Creación de un Daemon de Linux usando C y configuración del mismo como un servicio (Servidor Web)

Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

CE-4303 Principios de sistemas operativos

Bryan Abarca Weber
Raúl Arias Quesada
Rony Paniagua Chacón

I. Introducción

El presente documento tiene como finalidad presentar la solución a la tarea corta 1, la cual consiste en la creación de un web server en el lenguaje C y convertirlo en un servicio daemon en el sistema operativo Linux.

Un daemon o demonio es un proceso que se ejecuta en segundo plano, no se conecta a ninguna terminal de control. Estos normalmente se inician en el momento del arranque mediante scripts de SysV init que se encuentran comúnmente en /etc/init.d o /etc/rc.d o con scripts de systemd (dependiendo del sistema). Cuando la máquina bootea se corre el primer proceso (init o systemd) el cual se encarga de levantar todos los daemons (que a su vez estos pueden levantar otros daemons) que deben ejecutarse al comienzo del funcionamiento del sistema. Pero no todos son configurados para iniciar en la secuencia de booteo, si no que también pueden ser iniciados manualmente por el usuario desde la consola de comandos del sistema, o por otros daemons que bajo cierta configuración se ejecutan en determinado momento. Los daemons se ejecutan como root o algún otro usuario especial (como apache o postfix), y se encargan de las tareas a nivel del sistema [1].
Estos tienen dos requerimientos generales: tienen que correr como un hijo de init (primer proceso que se ejecuta en Linux) y no deben estar conectados a la terminal [1].
En [1] se describen los siguientes pasos para que un programa se convierta en un Daemon:

1. Llama a fork (). Esto crea un nuevo proceso, que se convertirá en el demonio.
2. Cambia le file mode mask (umask) por si necesita escribir en cualquier archivo (incluidos los registros) creados por el demonio y así asegurarse de que puedan escribirse o leerse correctamente.
3. En el padre, llama a exit (). Esto garantiza que el padre original (el abuelo del daemon) esté satisfecho de que su hijo haya terminado, que el padre del daemon ya no se esté ejecutando y que el daemon no sea un líder de grupo de procesos. Este último punto es un requisito para la finalización exitosa del siguiente paso.
4. Llama a setsid (), dando al demonio un nuevo proceso de grupo y sesión, los cuales lo tienen como líder. Esto también garantiza que el proceso no tenga un terminal de control asociado (ya que el proceso acaba de crear una nueva sesión y no asignará uno).
5. Cambia el directorio de trabajo al directorio raíz a través de chdir (). Esto se hace porque el directorio de trabajo heredado puede estar en cualquier parte del sistema de archivos. Los demonios tienden a ejecutarse durante el tiempo de actividad del sistema, y no desea mantener abierto un directorio aleatorio y, por lo tanto, evitar que un administrador desmonte el sistema de archivos que contiene ese directorio.
6. Cierra todos los descriptores de archivos. Ya que no se desea heredar descriptores de archivos abiertos y, sin saberlo, mantenerlos abiertos.
7. Abre los descriptores de archivo 0, 1 y 2 (standard in, standard out y standard error) y los redirige a /dev/null.

En muchos sistemas de Linux estos pasos son automatizados con funciones propias del sistema. Por ejemplo la función daemon(), que realiza este proceso.

Los daemons son utilizados con gran variedad de objetivos, por mencionar los más conocidos [3]:

1. Existen daemons como servicios que están escuchando solicitudes en puertos, como el www daemon; o para funcionalidades relacionadas con aplicaciones de internet, como el manejo de correos (sendmail daemon, finger daemon, talk daemon, etc.).
2. También son útiles para la administración del sistema, como estar verificando que no haya errores en el sistema, monitoreando el estado de los componentes (por ejemplo, temperatura), gestión de registros de archivos, limpieza diaria del sistema de archivos, administración de bases de datos, entre otros.
3. Otro empleo es para responder a solicitudes de archivos

por parte de los clientes o componentes del sistema (por ejemplo, los que se encargan de la asignación de puertos), o para programas que son necesarios que estén ejecutándose en todo momento, incluso cuando no se haya iniciado sesión con algún usuario.
4. Algunas partes del Kernel gestionan como si fueran procesos de usuarios, para lo que se emplean daemons. En su mayor parte estos procesos se encargan de aspectos con dispositivos de E/S, gestión de memoria y sincronización del caché de disco. Estos no pueden ser manipulados por el administrador del sistema, si no que se dejan por su propia cuenta.
5. Algunos daemons de vital importancia cumplen con funciones de alta responsabilidad, como la administrar otros daemons o servicios del sistema.
6. Los usuarios también pueden crear daemons para mantener programas corriendo con un objetivo específico sin necesidad de estar iniciándolos manualmente.

Con los administradores de servicios systemctl o service (los cuales también son daemons), dependiendo de cómo la distribución de Linux hace uso de systemd o init, es posible acceder a 4 principales funciones para los daemons [4]:

- start: comienza a ejecutarse el Daemon.
- stop: cuando el daemon se está ejecutando, es detenido.
- restart: cuando el daemon se está ejecutando, es detenido y luego es iniciado nuevamente.
- status: indica el estado del daemon, si este está ejecutándose.

En cuanto a su funcionamiento, es un programa que está corriendo constantemente en background, y se inicia automáticamente en el arranque de la máquina, para realizar alguna acción, por ejemplo, estar pendiente de solicitudes de recursos o algún servicio, como los recursos web.
Sin embargo, hay daemons que se ejecutan solo por algún tiempo, por ejemplo, el cron daemon, el cual se ejecuta una vez cada minuto, valida si hay alguna acción que deba realizar, si es así ejecuta el trabajo, si no vuelve a "dormir" hasta que sea tiempo para la siguiente validación [2].
Hay algunas convenciones comunes que son seguidas por daemons en el sistema UNIX [5].

- Las opciones de configuración generalmente se almacenan en /etc: El archivo de configuración se llama name.conf, donde nombre es el nombre del daemon o el servicio. Por ejemplo, la configuración para el syslogd daemon suele ser /etc/syslog.conf.
- Si un daemon tiene un archivo de configuración, el daemon lee el archivo cuando se inicia y no lo vuelve a ver. Si un administrador cambia la configuración, el daemon deberá detenerse y reiniciarse para tener en cuenta los cambios de configuración.

El servidor web debe ejecutarse como un servicio de Linux, esto se puede lograr utilizando el sistema de init SysVinit o Systemd. A continuación se presentan las principales características de estos.

Systemd
El conjunto de herramientas de systemd proporciona un modelo de inicio rápido y flexible para administrar una máquina completa desde el boot en adelante. //Algunas características de Systemd se listan a continuación:

1. Systemd proporciona capacidades de paralelización agresivas.
2. Utiliza la activación de socket y D-Bus para iniciar los servicios.
3. Ofertas a pedido de arranque de demonios.
4. Realiza un seguimiento de los procesos utilizando Linux cgroups.
5. Soporta snapshotting y restauración del estado del sistema
6. Mantiene los puntos de montaje y automontaje.
7. Implementa una lógica de control de servicio basada en dependencia transaccional elaborada

El objeto básico que systemd maneja y sobre el que actúa es una unidad. Las unidades pueden ser de muchos tipos, pero el tipo más común es un servicio (indicado por un archivo de unidad que termina en .service). Para administrar los servicios, en un servidor habilitado para systemd, la herramienta principal es el comando systemctl [6].
Systemd tiene un concepto de objetivos (targets) que tienen un propósito similar a los niveles de ejecución pero que actúan de manera un poco diferente [8]. De [7] se obtiene la información del siguiente cuadro:

Cuadro I
SYSTEMD TARGETS

Objetivo (target)	Descripción
poweroff	Apagar (entrar en el nivel de ejecución 0)
rescue	Entrar en modo de rescate (nivel de ejecución 1, usuario único)
multi-user	Entrar en modo multiusuario (niveles de ejecución 2 a 4)
graphical	Entrar en modo gráfico (nivel de ejecución 5)
reboot	Reiniciar el sistema (nivel de ejecución 6)

Todos los comandos normales del sistema init tienen acciones equivalentes con el comando systemctl. Para demostrar los comando utilizados por systemd se toma como ejemplo el nombre del servicio que se crea en la presente tarea, WebServer [6]. De [6] se obtienen los siguientes comandos.

- Para iniciar manualmente el servicio se utiliza el comando:
  systemctl start WebServer
- Para detener el servicio se utiliza el comando: stop WebServer
- Para reiniciar el servicio se ejecuta: systemctl restart WebServer
- Para intentar recargar el servicio sin interrumpir la funcionalidad normal, se ejecuta: systemctl reload WebServer
  Por defecto, la mayoría de los archivos de unidades del sistema no se inician automáticamente en el arranque. Para configurar esta funcionalidad, debe "habilitar.ª la unidad. Esto lo conecta a un cierto .ºbjetivo"de arranque,

lo que hace que se active cuando se inicie ese objetivo [6].

- Para permitir que un servicio se inicie automáticamente en el arranque, se ejecuta: systemctl enable WebServer
- Si desea volver a desactivar el servicio. systemctl disable WebServer
  Con systemd se puede tener acceso al estado general del sistema y también se puede obtener información sobre el estado de las unidades individuales.
- Para obtener una visión general del estado del sistema se ejecuta (unidades que systemd ha listado como .ªctivos"): systemctl list-units
- Para ver un resumen del estado actual de una unidad systemctl status WebServer

II. Ambiente de desarrollo utilizado

Para el desarrollo de este proyecto fueron necesarias las herramientas:

1. Sistema operativo con distribución LInux preferiblemente Ubuntu.
2. Editor de código C, incluso puede ser utilizado el editor de texto del sistema operativo, en nuestro caso utilizamos el editor libre Visual Studio Code.

III. Estructuras, funciones y librerías:

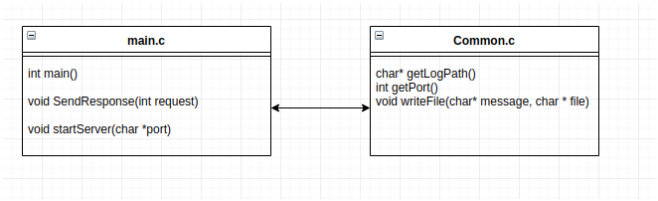

Figura 1. Diagrama de la solución.

El proyecto cuenta con dos archivos el main.c donde se instancia y crea el web server y el archivo Common,c con funciones comunes para la lectura y escritura de datos en un archivo, basado en [11].
En el archivo main se encuentran los métodos:

1. startServer: este método recibe como entrada el puerto, se encarga de levantar el servidor web, ejecutando las acciones socket y bind del servidor, esto utilizando la librería sys/socket.h.
2. SendResponse: Recibe como entrada el request a contestar, se encarga de verificar el request y responder utilizando HTTP-1.1 para ello ejecuta métodos como recv para leer el request y write para responder.
3. main: Método principal encargado de leer los valores de los archivos de configuración, levantar el servidor y se mantiene escuchando y aceptando conexiones est utilizando los otros métodos creados.

En el archivo Common se encuentran los métodos:

1. getLogPath: Método que lee el archivo de configuración y devuelve el path para el archivo donde escribir los logs, para ello busca la linea que tenga el valor de "LOGFILE", en caso de no encontrar el valor del path devuelve el valor por defecto /var/log/syslog. Utiliza métodos como fopen para abrir el archivo, fgets para leer líneas del archivo y sscanf para buscar la línea correcta.
2. getPort: Método que lee el archivo de configuración y devuelve el puerto a utilizar, para ello busca la linea que tenga el valor de "PORT", en caso de no encontrar el valor del puerto devuelve el valor por defecto 8081. Utiliza métodos como fopen para abrir el archivo, fgets para leer líneas del archivo y sscanf para buscar la línea correcta.
3. write: Recibe los datos a escribir y la ruta del archivo donde escribir, además de escribir el mensaje requerido escribe la hora del sistema para tener un mejor log de la actividad. Utiliza la librería time.h para obtener la hora y métodos como fopen para abrir el archivo y fprintf para escribir en él.

IV. Instrucciones de uso

Pasos previos a compilar el proyecto:

1. Cambiar la ruta donde se ubicará el archivo ejecutable, esto modificando el archivo WebServer.service, dentro del archivo modificar el parámetro ExecStart, por la ruta donde se encuentra la carpeta WebServerDir.
2. Dar permisos de escritura al archivo /var/log/syslog en caso de utilizar el archivo por defecto.
3. Ajustar el archivo config.conf con el valor del puerto y archivo donde registrar el log.

Compilar la tarea con el comando make desde una terminal ubicada en la ruta donde está el archivo Makefile entregado. Con esto el servicio estará funcionando.
Para utilizar el servicio se deben pegar los archivos a mostrar en la carpeta WebServerDir y podrán ser accesados desde un navegador web en la ruta localhost:puerto/nombreArchivo/extensión

Para interactuar con el servicio se pueden usar los comandos de una terminal: sudo systemctl start WebServer: Para iniciar el servicio. sudo systemctl stop WebServer: Para detener el servicio. sudo systemctl restart WebServer: Para reiniciar el servicio. sudo systemctl status WebServer: Para obtener el estado del servicio

Adicionalmente se brinda el archivo UnistallWebServer.sh para remover el servicio, el cual se puede utilizar ejecutando el comando: sh UnistallWebServer.sh esto desde una terminal en la ubicación del archivo.