Área Académica de Ingeniería en Computadores

Programa de Licenciatura en Ingeniería en Computadores

Curso: CE-4303 - Principios de Sistemas Operativos

Evaluación: Tarea Corta 1

Realizado por:

Daniel Canessa Valverde, 201137483

Felipe Mejías Loría, 201231682

Edward Umaña Williams, 201128403

Profesora:

Jennifer Vargas

Fecha: 16 de agosto de 2016

1 Introducción

1.1 Demonios en Linux

Un demonio es un tipo de programa de Unix que se ejecuta de manera constante sin que el usuario se de cuenta. Muchos de estos demonios se ejecutan constantemente esperando algún evento significativo. Otros demonios se ejecutan de manera periódica y concluyen al ejecutar su tarea. Por ejemplos el uucp, utilidad de transferencia de archivos. [1]

Los demonios son normalmente inicializados como un proceso. Un proceso es una instancia existente de un programa, la cual es administrada por el kernel a la que se le asigna un identificador de proceso. Existen tres tipos de procesos en linux: interactivos, batch y demonios. [2]

Los interactivos se ejecutan a nivel de usuario mediante línea de comandos y los bash se agregan a una cola de comandos que no está asociada a la línea de comandos utilizada. Por su parte los demonios son reconocidos por el sistema como un proceso normal, aunque no esté en contacto con el usuario y normalmente se inician con el boot del sistema

Los demonios son muy sencillos de programar en el ambiente the UNIX. Pueden ser escritos por cualquier usuario y lanzado periódicamente mediante la línea de comandos, o System V.

Para crear un demonio se deben seguir lo siguientes pasos: [3]

1. fork() para que el proceso padre pueda salir, así la devuelve control a la línea de comandos que llamó al programa.
2. setsid() para hacerse una sesión y una sesión de grupo lider de manera tal que el proceso no tenga una terminal controlada.
3. fork() de nuevo para que el padre (la sesión del grupo líder) pueda salirse. Esto significa que estamos en una sesión sin grupo leader, por lo que no puede controlar de nuevo una terminal.
4. chdir("/") para asegurarse que nuestro proceso no retenga ningún directorio en uso.
5. unmask(0) para tener completo control the permisos sobre cualquier archivo que escriba. close() fds 0, 1 and 2. Libera los estandar de entrada , salida y error heredados del proceso padre.
6. Establecer un archivo para registrar los eventos y errores del demonio

Los sistemas de Unix ejecutan una gran cantidad de demonios, principalmente para responder a peticiones de servicios de computadores en la red, además de responder a otros programas y actividades del hardware. No es necesario que el responsable de ejecutar la acción del demonio este conciente que el demonio está escuchando aunque algunos programas frecuentemente ejecutan acciones porque saben que un demonio está atento a esas señales. [2]

Dentro de los demonios más importante de Linux , y por lo tanto las funciones más importantes de los mismo podemos encontrar init, el programa encargado de lanzar todos los otros procesos, crond , el calendarizador de tareas basado en tiempo o por ejemplo httpd que es un servidor web entre otros como ntpd,sshd, syncd entre otras. [4]

1.2 SystemD y SysV

System V

System V fue una de las versiones del sistema operativo Unix. System V es la fuente de varias características comunes de Unix, tales como los "SysV init scripts", ubicados en /etc/init.d; usados para el control de inicio y apagado del sistema. El sistema de inicio System V controla el arranque de los programas en el instante de inicio de la máquina. Este sistema de inicio es considerado por muchos más fácil de usar, más potente y flexible que el sistema de inicio BSD tradicional. El proceso Init es un programa que el núcleo ejecuta cuando arranca el sistema. Se encarga de inicializar todos los procesos normales que se necesiten ejecutar en el momento de arrancar; incluyendo los terminales que le permiten acceder al sistema y cualquier proceso que quiera ejecutar cuando su máquina arranque. [5]

¿Cómo funciona los métodos de System V?

Con System V, la mayoría de las aplicaciones estándar que se pueden instalar se iniciarán al iniciar el sistema por defecto, Cada una de estas aplicaciones tiene sus propios scripts de inicio en /etc/init.d. Para aplicaciones personalizadas, el usuario tiene que crear sus propios scripts de inicio y habilitar los servicios para que se inicie automáticamente. System V utiliza Bash para los scripts de inicio. Para que un demonio se inicie o se detenga automáticamente cuando iniciamos y apagamos el sistema respectivamente debemos asignar los runlevels. El arranque

de un sistema operativo UNIX tipo System V se puede dividir en dos etapas principales: el arranque del núcleo del sistema operativo y la inicialización que lleva acabo el programa "init". Un nivel de ejecución representa el estado actual de un sistema Linux. Cada nivel de ejecución indica qué servicios pueden estar en ejecución en ese estado. Algunos servicios pueden ejecutarse en uno o más niveles de ejecución, pero no en otros niveles. Los niveles de ejecución se denotan por un solo dígito y pueden tener un valor entre 0 y 6. La siguiente lista muestra cada uno de los niveles de ejecución: [5]

- Nivel de ejecución 0: Apagado del sistema.
- Nivel de ejecución 1: Único usuario, modo de rescate.
- Nivel de ejecución 2: Multi-usuario, modo de texto con funciones de red habilitadas.
- Nivel de ejecución 3: Multi-usuario, modo de texto con funciones de red habilitadas.
- Nivel de ejecución 4: Multi-usuario, modo de texto con funciones de red habilitadas.
- Nivel de ejecución 5: Multi-usuario, red habilitada, en modo gráfico.
- Nivel de ejecución 6: Reinicio del sistema.

Los niveles de ejecución 2,3 y 4 varían dependiendo de la distribución. En System V, el sistema operativo se iniciará con un nivel de ejecución en particular; y, cuando se inicia, se tratará de iniciar todos los servicios que están asociados con ese nivel de ejecución. La secuencia de arranque de System V es:

- El demonio init se crea a partir del archivo binario /sbin/init.
- El primer archivo que el demonio init lee es /etc/inittab.
- Una de las entradas de este archivo decide el nivel de ejecución en el cual la máquina debe arrancar.
- Luego el demonio init sigue leyendo en el archivo /etc/inittab y lee lo que el script init necesita para funcionar en ese nivel de ejecución.

Así, cuando el demonio init encuentra los init scripts que se necesitan para funcionar en el nivel dado, es esencial averiguar cuáles son los servicios que necesita poner en marcha. Estos son los scripts de inicio donde se puede configurar el comportamiento de inicio para los servicios personalizados. [5]

¿Cómo son implementados los métodos?

Un script de inicio es lo que controla un servicio específico, como Trackermon, en System V. Los scripts de inicio para los servicios son proporcionados por el proveedor de la aplicación o vienen con la distribución de Linux. También podemos crear nuestros propios scripts de inicio para los servicios personalizados, en este caso para el servicio Trackermon. Cuando un proceso o servicio como Trackermon se inicia, su archivo de programa binario tiene que cargarse en la memoria. Dependiendo de cómo esté configurado el servicio, este programa puede tener que seguir ejecutandose en segundo plano de forma continua. La tarea de iniciar, detener o volver a cargar esta aplicación binaria está a cargo de los scripts de inicio del servicio. Se llama el script de inicio, ya que inicia el servicio. El directorio /etc es el directorio padre para los scripts de inicio. La ubicación real de los scripts de inicio está dentro de /etc/init.d. Estos scripts son un enlace simbólico a los directorios rc. Dentro del directorio /etc, hay una serie de directorios rc, cada uno con un número en su nombre. Los números representan diferentes niveles de ejecución. Así que para implementar un servicio en System V, se debe crear un script que contenga los métodos: start, stop, restart y status. Éste script se debe colocar en la carpeta /etc/init.d, de esta forma se podrá configurar como un servicio de Linux. [5]

Systemd

Systemd es un marco completamente nuevo que abarca muchos componentes de un sistema Linux moderno. Una de sus funciones es la de trabajar como un sistema y gestor de servicios para Linux. En esta función, uno de las cosas que Systemd controla es como un servicio debe comportarse si se bloquea o se reinicia el equipo. Systemd es compatible con los comandos de System V y sus scripts de inicialización. Eso significa que cualquier servicio de System V también se ejecutará en systemd. [5]

¿Cómo funciona los métodos de Systemd?

En systemd los servicios se denominan units. Cada unit se define en un archivo donde se especifica un proceso para arrancar por systemd. La principal diferencia entre systemd y system v, es que systemd es responsable de la inicialización, no sólo de los demonios de servicios, sino también de otros tipos de recursos como las rutas del sistema operativo del dispositivo, puntos de montaje, enchufes, etc. Un recurso puede ser cualquiera de estos. Los archivos que definen los units se pueden encontrar básicamente en tres ubicaciones distintas: [6]

- /usr/lib/systemd/system/: unidades distribuidas con paquetes RPM instalados.
- /run/systemd/system/: unidades creadas en tiempo de ejecución.
- /etc/systemd/system/: unidades creadas y administradas por el administrador del sistema.

El formato de un archivo unit sigue un conjunto de reglas específicas. El archivo se divide en las siguientes secciones:

- Unit
- Service
- Install

Dentro de cada una de estas secciones se pueden especificar los siguientes parámetros. [6]

Sección Unit

- Description=Se indica una descripción del servicio que se muestra al consultar el status del servicio.
- After= Se indica el orden en el cual los units se inician. El unit se inicia sólo después de que los units especificados en esta línea estén activos.
- Requires= Aquí se indica la dependencias sobre otros units. Los units listados aquí serán activados junto con este unit. Si alguno de los units requeridos falla en el arranque, este unit tampoco se activa.

- Wants= Activa los units indicados aquí. Wants configura dependencias de manera más débil que Require. Si alguno de los units indicados por Wants no se inician correctamente no tienen ningún efecto en el estado de este unit. Wants es la manera recomendada para establecer dependencias personalizadas.
- Conflicts= Configura dependencias negativas, es decir, es un opuesto a Requires. El servicio no se inicia si el servicio indicado en esta línea está activo.

Sección Service

- TimeoutStartSec= Tiempo tras el cuál, si el servicio no ha arrancado, se considera fallo y se detiene.
- ExecStart=comando a ejecutar.
- Type=Configura el tipo de arranque del procesos de la unidad la cual afecta a la funcionalidad ExecStart. Las opciones son:
  - simple: El proceso arrancado con ExecStart es el proceso principal del servicio. Este proceso se arranca inmediatamente. No utilizar este tipo si otros servicios necesitan ejecutarse en orden con él.
  - forking: El proceso iniciado con ExecStart genera un proceso hijo que se convierte en el proceso principal del servicio. Se sale del proceso padre cuando el arranque se completa. El uso de esta opción es importante cuando ejecutamos un script que a su vez ejecuta otros procesos.

Sección Install

- WantedBy=runlevel.target Indica el target al que pertenece este unit. Lo que se consigue con esto es que el servicio se ejecute automáticamente al arrancar el target especificado.

Un conjunto de units definen un target. En systemd, un target representa un runlevel. La siguiente lista muestra los niveles de ejecución en systemd: [5]

- Nivel de ejecución 0: poweroff.target

- Nivel de ejecución 1: rescue.target
- Nivel de ejecución 2: multi-user.target
- Nivel de ejecución 3: multi-user.target
- Nivel de ejecución 4: multi-user.target
- Nivel de ejecución 5: graphical.target
- Nivel de ejecución 6: reboot.target

En Systemd la forma de controlar los servicios del sistema es diferente a System V. Los servicios ya no se controlan a través de /etc/init.d y tampoco se utiliza el comando "service". En Systemd se utiliza el gestor de servicios llamado systemctl. La principal orden para controlar systemd es systemctl. Systemctl sustituye a chkconfig de System V. Para la gestión de servicios en Systemd se utilizan los siguientes comandos: [6]

- systemctl start <NombreServicio>.service: Se utiliza para iniciar el servicio.
- systemctl stop <NombreServicio>.service: Se utiliza para parar el servicio.
- systemctl status <NombreServicio>.service: Se utiliza para visualizar el estado de ejecución de un servicio.
- systemctl is-active <NombreServicio>.service: Se utiliza para ver si el servicio está activo.
- systemctl enable <NombreServicio>.service: Se utiliza para habilitar servicios en el arranque.
- systemctl disable <NombreServicio>.service: Se utiliza para deshabilitar servicios en el arranque.
- systemctl restart <NombreServicio>.service: Se utiliza para reiniciar un servicio.

¿Cómo son implementados los métodos?

Para implementar un servicio en Systemd, lo que se tiene que hacer es lo siguiente: Primero que todo, se debe crear un archivo.service, en el cual se especifica el nivel de ejecución del servicio, la descripción del servicio y el programa que debe ejecutar el servicio cuando éste se inicializa. Luego de crear este archivo, se debe colocar en la carpeta /etc/systemd/system/, de esta forma ya se puede inicializar manualmente el servicio en la terminal.

1.3 Especificación del programa

Se debe crear un demonio en Linux y debe ser configurado como un servicio utilizando un init script. El servicio Trackermon se ocupará de vigilar los recursos importantes a nivel del sistema operativo como memoria, CPU y archivos del sistema, y anotar las alertas en el archivo log especificado. Trackermon debe ser creado en C para Linux. El programa debe cumplir con los siguientes requerimientos:

1. El demonio Trackermon se va a encargar de:
   (a) Supervisar el uso del CPU y generar una alerta que se añadirá en el archivo log cuando el umbral(threshold) del CPU es igual o mayor que lo que se especifica en el archivo de configuración.
   (b) Supervisar el uso de la memoria y generar una alerta que se añadirá en el archivo log cuando el umbral(threshold) del uso de la memoria es igual o mayor que lo que se especifica en el archivo de configuración.
   (c) Supervisar los archivos del sistema y generar una alerta que se añadirá en el archivo log cuando el umbral(threshold) de los archivos del sistema es igual o mayor que lo que se especifica en el archivo de configuración.
   (d) Leer el archivo de configuración para cargar los umbrales(thresholds) para la supervisión de los recursos, la ubicación de este archivo de configuración va a ser /etc/trackermon/config.conf. Este archivo tiene que ser recargado en el sistema cada vez que el demonio se inicia por primera vez o se reinicia.
2. Trackermon debe correr como un servicio de Linux. Con la creación del init script, se van a implementar dos formas de iniciar o parar el demonio:
   (a) Inicializar Trackermon en tiempo de arranque.

(b) Parar Trackermon cuando el sistema operativo se apague.

(c) Iniciar, parar, consultar el estado y reinicio de Trackermon: se puede hacer desde la terminal como root.

2 Ambiente de desarrollo

- El software se desarrolló en el lenguaje de programación C, utilizando el ambiente de desarrollo Geany versión 1.28.
- El sistema operativo que se utilizó fue Linux y su distribucción Ubuntu 15.04.
- Para compilar el código se utiliza el compilador GCC, con la versión 6.1.

3 Estructuras de datos, funciones y librerias

El desarrollo de la aplicación se dividió en 2 partes, el desarrollo del demonio de monitoreo y la implementación del servicio. El demonio cuenta con 2 archivos de código en C, que se incluyen en un tercer archivo de código C que es el encargado de unirlos. Es así como el demonio cuenta con un código dedicado a la lectura del archivo config.conf (readFile), otro código dedicado al monitoreo (monitor) y otro código que genera al demonio y une los dos códigos anteriores (TrackerMon).

El archivo de código principal, TrackerMon, está compuesto de 2 funciones skeleton_daemon y main, a continuación se explican con más detalle.

- main: es la función principal de este código, cuando el código se ejecuta está función es la primera en ejecutarse. Esta función inicialmente hace una llamada a la función skeleton_daemon, lo que demoniza el código, luego de esto se inicia un ciclo infinito que tiene un delay de 10 segundos al finalizar cada iteración. El while primero obtiene los datos del archivo config.conf utilizando el código de readFile que se explica más adelante, readFile provee estos datos por medio de una estructura. Luego se procede a llamar a las funciones del monitor suministrandoles los datos obtenidos de ReadFile, específicamente isMemoryRangeAccepted, isCPURangeAccepted e isSystemFileRangeAccepted.
- skeleton_daemon: este método se encarga de crear un demonio siguiendo los pasos establecidos para este fin. Iniciando con un fork() donde el proceso padre termina y el hijo ignora las señales de finalización del proceso

padre y de la terminal, mediante los métodos signal(SIGCHLD, SIG_IGN) y signal(SIGHUP, SIG_IGN). De nuevo se hace un fork() y deja que el padre termine. Luego este tercer proceso hijo hace un umask(0) para obtener los permisos correspondientes al proceso sobre archivos, luego se cambia el directorio de trabajo utilizando chdir("/"). Y por último el mismo abre el archivo de log para confirmar el funcionamiento del demonio.

El código del archivo monitor.c es el encargado de realizar las verificaciones de los recursos del sistema que fueron especificados en el archivo config.conf , en caso de que se exceda el "threshold" de un recurso especificado en el archivo de configuración, el monitor escribe en Trackermon.log un reporte de la situación, este archivo por defecto se encuentra en /var/log/Trackermon.log, sin embargo esta ruta puede cambiar en el archivo de configuración. Las funciones del monitor se especifican a continuación.

- isMemoryRangeAccepted: esta función calcula la cantidad de memoria Ram que está en uso, en caso de superar el valor del "threshold" escribe en el archivo Trackermon.log un reporte.
- isCPURangeAccepted: esta función calcula la cantidad de uso del CPU actualmente, en caso de superar el valor del "threshold" escribe en el archivo Trackermon.log un reporte.
- isSystemFileRangeAccepted: esta función revisa el almacenamiento de cada system file especificado en el archivo config.conf, en caso de superar el valor del "threshold" de algún system file escribe en el archivo Trackermon.log un reporte.

El monitor utiliza varias bibliotecas, estas se especifican a continuación:

- Unistd.h: es la biblioteca encargada de dar acceso al API de POXI. De los cuales se utilizan métodos como fork() para crear procesos hijos en la creación del demonio.
- Signal.h: esta biblioteca permite el manejo de señales que se reportan durante la ejecución del programa, de igual manera esta es importante para la generación del demonio y es utilizada en las siguientes líneas de código como signal(SIGCHLD, SIG_IGN) donde se le indica al proceso

que ignore la señal enviada por su padre cuando este termine el proceso y signal(SIGHUP, SIG_IGN) que le indica al proceso que ignore la señal de que la terminal donde fue ejecutado ha sido cerrada.

- sys/types.h y sys/stat: la librería sys/stat permite obtener información sobre ficheros y en sys/types están definidos algunos tipos de datos utilizados por la biblioteca anterior. Estos son importantes para leer archivos y obtener datos del monitoreo del CPU y memoria.

- syslog.h: es la biblioteca utilizada para el registro de mensajes. Utilizados en el proyecto para debug los mensajes de error de los procesos.

- sys/sysinfo.h: esta biblioteca devuelve estadísticas respecto al uso de la memoria y el uso de swap también como la carga promedio de la memoria y otros datos. Mediante el metodo int sysinfo(struct sysinfo *info) que almacenara todos los datos en la estructura ingresada como parametro.

- sys/statvfs.h: para calcular el uso de los archivos del sistema se utilizó esta biblioteca que brinda datos sobre los archivos del sistema que permiten el cálculo del uso actual de las carpetas del sistema. Mediante el metodo statvfs(path, &vfs) donde los datos de la carpeta indicada en el path seran guardados en el parametro vfs.

Por otro lado código del archivo readFile.c tiene como función leer el archivo de configuración config.conf que se encuentra ubicado en /etc/trackermon/config.conf, este archivo contiene la especificación de lo que se debe monitorear y los threshold de cada parámetro a monitorear. En readFile.c se hace uso de una estructura de datos denominada configFile, esta estructura consta de:

- configFile.CPUthresholdResult: este dato es un int, contiene el threshold especificado del CPU.

- configFile.MemthresholdResult: este dato es un int, contiene el threshold especificado de la memoria RAM.

- configFile.LogFileResult: este dato es un arreglo de char, contiene la dirección en la que se debe escribir el archivo .log.

- configFile.FileSystemDataResult: este dato es un arreglo bidimensional de char, contiene los nombres de los system file a monitorear.

- configFile.FileSystemValueResult: este dato es un arreglo de int, contiene los threshold de los system file.
- configFile.index: este dato es un int, representa la cantidad de system file especificados en el archivo de configuración.

Las funciones de readFile.c se especifican a continuación:

- findSubstr: esta función recibe un arreglo de char y verifica si este se encuentra en otro arreglo de char, en caso de ser afirmativo retorna la posición en la que se encuentra, en otro caso retorna -1.
- getConfigFileInfo: esta función lee el archivo de configuración, extrae los datos y los encapsula en una estructura del tipo configFile, esta estructura es retornada.

El servicio es el encargado de ejecutar el demonio automáticamente. El servicio se encuentra ubicado en /etc/systemd/system/Trackermon.service. Para que el servicio inicie en el run level 5, se especifica en el servicio que este debe ejecutarse después de "graphical.target". Además en el se especifica que debe ejecutar ExecStart=/usr/bin/TrackerMon.

Diagrama UML

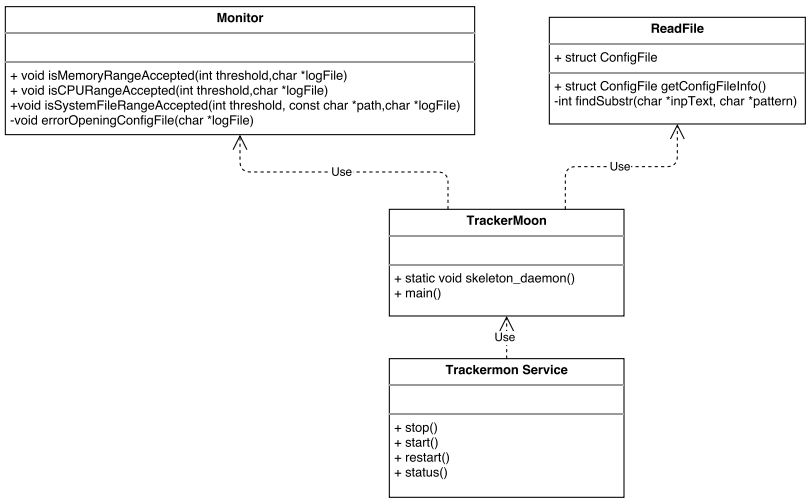

Figura 1. Diagrama UML del proyecto.

Intrucciones de como usar el programa

Para utilizar el programa Trackermon debe escribir los siguientes comandos en la terminal de Linux, se recomienda tener privilegios de usuario administrador, en otro caso sólo podra saber el estado de Trackermon.

- sudo service Trackermon start - este comando se utiliza iniciar el servicio Trackermon.
- sudo service Trackermon stop - este comando se utiliza detener el servicio Trackermon.
- sudo service Trackermon restart - este comando se utiliza reiniciar el servicio Trackermon.
- service Trackermon status - este comando se utiliza para saber el estado del servicio Trackermon.

Recuerde que en todo momento puede revisar el archivo Trackermon.log donde se encuentran todos los reportes del sistema Trackermon, para hacer esto escriba en la terminal /var/log/trackermon.log, además puede modificar el archivo config.conf para modificar los valores de los threshold y agregar system files a monitorear, para realizar esta acción escriba en la terminal sudo gedit /etc/trackermon/config.conf.

Bitácora

Bitácora de Daniel Canessa Valverde

Fecha	Horas	Trabajo Realizado
12-8-2016	0.5	Crear archivo .conf
12-8-2016	7	Leer archivos .conf en C y generar un struct
13-8-2016	7	Adecuar script de servicio a systemd y hacer que trabaje con el demonio
14-8-2016	4	Documentación

Bitácora de Felipe Alberto Mejías Loría

Fecha	Horas	Trabajo Realizado
11-8-2016	3	Investigación previa sobre demonios en Linux, systemd y systemv
12-8-2016	7	Investigación sobre la construcción de servicios con systemd y systemv
13-8-2016	7	Configuracion de Trackermon como servicio mediante systemd
15-8-2016	4	Documentación

Bitácora de Edward Umaña Williams

Fecha	Horas	Labores Realizadas
10/08/2016	4	Investigación sobre demonios y primer ejemplo
11/08/2016	2	Investigación lecturas sobre CPU y memoria
12/08/2016	6	Demonio y mediciones de CPU y memoria
13/08/2016	4	Unir lectura archivos, con monitoreo y demonio
14/08/2016	1	Documentación
15/08/2016	3	Documentación

Estado final del proyecto

El proyecto se ha finalizado de forma satisfactoria cumpliendo con todos los requerimientos del cliente.

El mayor problema que se presentó fue el de la unión del servicio con el demonio, esto porque la distribución de linux que se utilizó es systemd y se intentaba hacer que el sevicio se ejecutara como sys-v, corregido este problema la creación del servicio se realizó sin contratiempos.

Conclusiones

- El manejo de cadenas de texto en C no es trivial como en lenguajes de más alto nivel, se debe tener claro si se desea hacer que esta sea dinámica o no, en caso de que sea dinámica se debe liberar la memoria que se utilizó para esta.
- Los demonios son esenciales para realizar tareas en segundo planos que no requieran interacción con el usuario.
- Linux ofrece métodos o archivos para la lectura de datos respecto al hardware como el uso del CPU y memoria.
- Systemd ofrece una mejor manera de configurar un demonio como un servicio y con menor código que System V.

Recomendaciones

- Se debe buscar que para el manejo de cadenas de texto el código sea lo más reutilizable posible, si esto no se hace el código se vuelve complejo.
- Utilizar demonios para procesos que no requieran interacción con el usuario.

Referencias

[1] 2016. [Online]. Available: http://cjh.polyplex.org/software/daemon.pdf. [Accessed: 15- Aug- 2016].

[2]"Daemon Definition", Linfo.org, 2005. [Online]. Available: http://www.linfo.org/daemon.html.

[Accessed: 15- Aug- 2016].

[3]"Unix Programming Frequently Asked Questions - 1. Process Control", Web.archive.org.

[Online].Available:http://web.archive.org/web/20120914180018/http://www.steve.org.uk/Reference/Unix

/faq_2.html#SEC16. [Accessed: 15- Aug- 2016].

[4] A. Verma, Unix and shell programming. Bangalore: Firewall Media/Laxmi Publications Pvt. Ltd., 2008.

[5]"How To Configure a Linux Service to Start Automatically After a Crash or Reboot – Part 2: Reference | DigitalOcean", Digitalocean.com, 2015. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-configure-a-linux-service-to-start-automatically-after-a-crash-or-reboot-part-2-reference. [Accessed: 15- Aug- 2016].

[6]"Linux – El sistema de inicio Systemd – mundotelematico.com", Mundotelematico.com, 2015. [Online]. Available: http://www.mundotelematico.com/?p=461. [Accessed: 15- Aug- 2016].