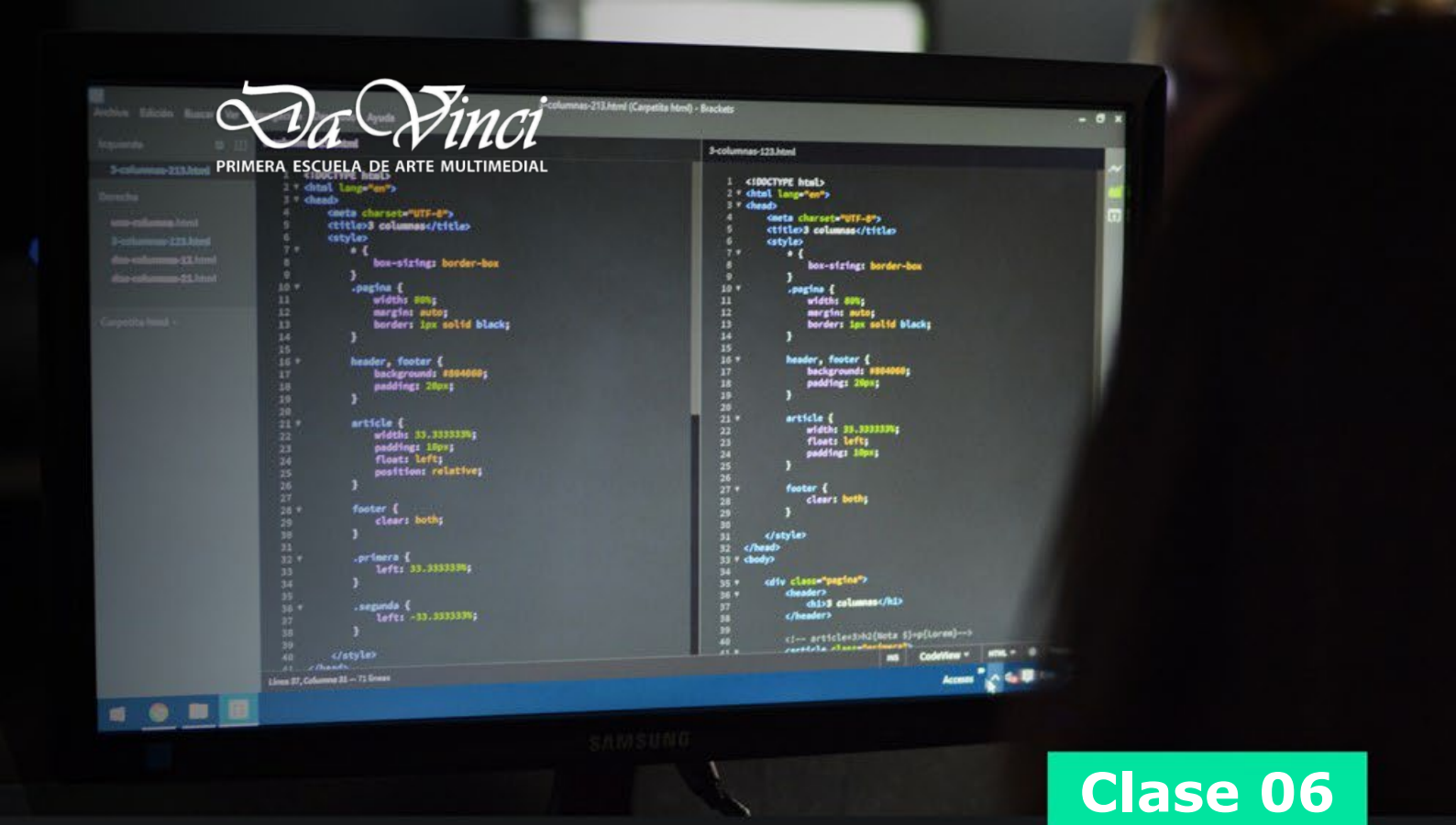

Clase 06

Diseño y Programación Web

Materia:

Sistemas Operativos

Docente contenidista: CARLASSARA, Fabrizio

Revisión: Coordinación

Contenido

Clase 6

¡Te damos la bienvenida a la materia **Sistemas Operativos**!

En esta clase vamos a ver los siguientes temas:

Vamos a comenzar a ver cómo administrar servicios, procesos y algunos comandos avanzados. Al finalizar la clase van a aprender:

- Administrar servicios en sistemas con SysV y systemd.
- Cómo ver y eliminar procesos.
- Cómo usar pipelines.
- Cómo redireccionar la salida de comandos.

Servicios

SysV Init y systemd son dos sistemas de inicio (init) utilizados en sistemas operativos basados en Linux para gestionar el proceso de arranque del sistema y la gestión de servicios. Ambos sistemas tienen diferentes enfoques y características, vamos a intentar ver un poco ambos y analizar algunas de sus diferencias:

SysV Init

SysV Init es uno de los sistemas de inicio más antiguos utilizados en sistemas operativos basados en Unix y Linux. Se basa en secuencias de comandos de inicio almacenadas en directorios específicos para iniciar y detener servicios durante el proceso de arranque y apagado del sistema. Algunas de sus funciones incluyen:

- Gestionar el proceso de arranque del sistema, iniciando servicios y procesos necesarios.
- Permitir iniciar y detener servicios de manera secuencial durante el inicio y apagado del sistema.
- Utilizar secuencias de comandos shell almacenadas en directorios como **/etc/init.d/** y **/etc/rcX.d/** para iniciar y detener servicios en diferentes niveles de ejecución (runlevels).

Los servicios se gestionan utilizando scripts de inicio ubicados en el directorio /etc/init.d/.

Los enlaces simbólicos a estos scripts de inicio se crean en los directorios /etc/rcX.d/, donde X es el nivel de ejecución del sistema (como 0 para apagar, 1 para modo de usuario único, etc.).

Los comandos **service** y **chkconfig** se utilizan para iniciar, detener y administrar servicios.

SysV Init todavía se encuentra en uso en algunas distribuciones más antiguas y sistemas embebidos.

Administrar servicios con SysV Init

Supongamos que queremos iniciar y detener el servicio *apache2* en un sistema que utiliza SysV Init. Podemos escribir el comando: **sudo service apache2 start**.

Este comando inicia el servicio *apache2* utilizando el script de inicio correspondiente ubicado en */etc/init.d/*.

Para detener el servicio, usamos el comando: **sudo service apache2 stop**.

Este comando detiene el servicio *apache2* utilizando el script de inicio correspondiente ubicado en */etc/init.d/*.

Podemos usar *chkconfig* también para configurar servicios y daemons para que se inicien automáticamente durante el arranque del sistema en determinados runlevels.

Siguiendo con el ejemplo de apache2, podemos:

- Ver el estado de un servicio con **chkconfig --list apache2**.
- Habilitar un servicio para que se inicie automáticamente con **chkconfig apache2 on**.
- Deshabilitar un servicio para que no se inicie automáticamente con **chkconfig apache2 off**.

systemd

Por otro lado, systemd es un sistema de inicio más moderno y más complejo que SysV Init. Se basa en un diseño modular y utiliza unidades de servicio y otras unidades para la gestión del sistema y los servicios. Entre sus funciones podemos mencionar:

- Gestionar el proceso de arranque del sistema y la gestión de servicios y procesos.
- Proporcionar un sistema de gestión de servicios más avanzado que permite la paralelización de la inicialización y una gestión más precisa de las dependencias entre servicios.
- Utilizar archivos de configuración llamados unidades (unit files) para definir cómo se inician y gestionan los servicios.

Las unidades de servicio se definen en archivos de configuración ubicados en directorios como **/etc/systemd/system/** y **/lib/systemd/system/**.

El comando **systemctl** se utiliza para administrar servicios, como iniciar, detener, reiniciar y habilitar/deshabilitar servicios. También puede utilizarse para ver el estado de los servicios y revisar los registros del sistema.

Administrar servicios con systemd

Supongamos que queremos administrar el servicio *apache2* en un sistema que utiliza systemd. Podemos:

- Iniciarlo con: **systemctl start apache2**.
- Detenerlo de servicios: **systemctl stop apache2**.
- Reinicio de servicios: **systemctl restart apache2**.
- Habilitarlo para que inicie con el sistema: **systemctl enable apache2**.
- Deshabilitar para que no inicie con el sistema: **systemctl disable apache2**.
- Ver el estado: **systemctl status apache2**.

Estos comandos trabajan con el servicio apache2 utilizando la unidad de servicio correspondiente ubicada en */lib/systemd/system/* o */etc/systemd/system/*.

Tanto para sistemas con SysV o con systemd se requieren tener privilegios de superusuario (sudo) para iniciar o detener servicios.

Daemons

Un daemon es un tipo de proceso que se ejecuta en segundo plano en sistemas operativos tipo Unix y Linux. Estos procesos no requieren interacción directa con el usuario y generalmente proporcionan servicios específicos al sistema operativo o a otras aplicaciones.

El init daemon, o init system, es el primer proceso que se inicia durante el arranque del sistema. Es responsable de iniciar y detener otros procesos y servicios durante el proceso de arranque y apagado del sistema. El init daemon también administra los runlevels, que son diferentes estados en los que puede encontrarse el sistema.

Runlevels

Los runlevels son diferentes estados en los que puede encontrarse un sistema Unix o Linux. Cada runlevel define qué servicios y procesos se ejecutarán en ese estado. Los runlevels varían según la distribución de Linux, pero los más comunes son:

Runlevel	Nombre	Descripción
0	Suspendido (Halt)	El sistema no tiene daemons activos en memoria y está listo para apagarse.
1 o S	Modo de usuario único	El sistema sólo tiene daemons suficientes para permitir que el usuario root se loguee y haga tareas de mantenimiento.
2	Modo multiusuario	El sistema tiene la mayoría de los daemons corriendo y acepta que múltiples usuarios se logueen y usen esos servicios.
3	Modo multiusuario extendido	Un sistema como el anterior pero que agrega servicios especiales de redes.
4	Sin uso	No se usa normalmente, puede ser personalizado.
5	Modo gráfico	Igual que el modo multiusuario extendido pero tiene un programa de login gráfico, normalmente corriendo en tty1 o tty7.
6	Reinicio	Usado para reiniciar el sistema.

Los administradores de sistemas pueden cambiar entre runlevels utilizando el comando **telinit** o **init**. Por ejemplo, para cambiar al runlevel 3, se puede ejecutar: **sudo telinit 3**.

Podemos ver el runlevel actual con el comando **runlevel**.

Los daemons se alojan en diferentes ubicaciones del sistema de archivos, pero generalmente los ejecutables se encuentran en directorios como **/usr/sbin/** o **/usr/bin/**, mientras que los archivos de configuración pueden estar en **/etc/**.

Scripts de configuración en tiempo de ejecución

Para ayudar a gestionar el uso de daemons, aparecen los scripts de configuración en tiempo de ejecución, también conocidos como

scripts de inicialización o init scripts. Estos son secuencias de comandos que se ejecutan durante el inicio o apagado del sistema para iniciar, detener o reiniciar daemons y servicios. Estos funcionan ligeramente distinto de acuerdo a si el sistema funciona con SysV o systemd.

Sistemas con SysV

En este tipo de sistema, se alojan directorios dentro de **/etc** con el nombre **rc[runlevel].d**, donde se guardan scripts de ejecución que el sistema operativo tiene que ejecutar cuando entra en el runlevel correspondiente. Por ejemplo, el directorio */etc/rc5.d/* es un directorio utilizado para especificar los servicios que se deben iniciar o detener cuando el sistema está en el runlevel 5.

Los nombres de los archivos en /etc/rc5.d/ están estandarizados y siguen una convención específica para indicar el orden de inicio o detención de los servicios durante el inicio o apagado del sistema. Estos nombres de archivo generalmente comienzan con una letra que indica la acción a realizar (iniciar o detener el servicio) seguida de un número y el nombre del servicio al que se refiere. Acá dejamos una explicación detallada de la convención de nombres:

- **S (Start):** Los archivos que comienzan con la letra "S" indican servicios que se deben iniciar durante el inicio del sistema en el runlevel 5.
- **K (Kill):** Los archivos que comienzan con la letra "K" indican servicios que se deben detener durante el apagado del sistema en el runlevel 5.
- **Número:** Después de la letra inicial, sigue un número que indica el orden en el que se iniciarán o detendrán los servicios. Los servicios con números más bajos se iniciarán o detendrán antes que los servicios con números más altos.
- **Nombre del servicio:** Después del número, se incluye el nombre del servicio al que se refiere el archivo. Esto puede ser el nombre real del servicio o un enlace simbólico al script de inicio o parada del servicio ubicado en el directorio /etc/init.d/.

Por ejemplo, supongamos que tienes un servicio llamado *apache2*. Los archivos en /etc/rc5.d/ relacionados con este servicio podrían ser:

- **S50apache2:** Este archivo indica que el servicio apache2 debe iniciarse durante el inicio del sistema en el runlevel 5 y se encuentra en la posición 50 en la secuencia de inicio de servicios.

- **K20apache2:** Este archivo indica que el servicio apache2 debe detenerse durante el apagado del sistema en el runlevel 5 y se encuentra en la posición 20 en la secuencia de detención de servicios.

Se puede escribir comandos de shell en el archivo **rc.local** para que se ejecuten luego de que el sistema operativo termine de inicializar los servicios correspondientes al runlevel en el que se encuentra.

Sistemas con systemd

En sistemas que utilizan systemd como sistema de inicio, la inicialización de scripts de configuración de daemons se gestiona de manera diferente a los sistemas que usan SysV Init. Los sistemas con systemd introducen un enfoque más moderno y flexible para la gestión de servicios y daemons, utilizando unidades de servicio (unit files) en lugar de scripts de shell tradicionales. Algunos conceptos que tenemos que tener en cuenta para entender su funcionamiento son:

- **Definición de unidades de servicio:** En systemd, los servicios se definen mediante archivos de configuración llamados unidades de servicio (unit files). Estos archivos especifican cómo systemd debe gestionar un servicio específico, incluyendo cómo iniciarlo, detenerlo, reiniciarlo, etc.
- **Ubicación de los archivos:** Las unidades de servicio se encuentran típicamente en los directorios */lib/systemd/system/* para las unidades proporcionadas por el sistema y */etc/systemd/system/* para las unidades de usuario (creadas por el administrador del sistema).
- **Formato de los archivos:** Las unidades de servicio son archivos de texto plano que siguen un formato específico de systemd. Estos archivos pueden contener secciones como *[Unit]*, *[Service]*, *[Install]*, entre otras, que definen diferentes aspectos del servicio, como dependencias, comandos de inicio, entornos, etc.

Cuando el sistema está iniciando, lo siguiente comienza a ocurrir para correr los daemons:

1. **Análisis de dependencias:** Durante el inicio del sistema, systemd analiza las dependencias entre los servicios y determina el orden en el que deben iniciarse.
2. **Inicio de servicios:** systemd inicia los servicios según lo especificado en las unidades de servicio. Utiliza paralelización y otras técnicas para optimizar el proceso de inicio y reducir el tiempo de arranque del sistema.

3. **Control y supervisión:** systemd supervisa continuamente el estado de los servicios y puede reiniciar automáticamente los servicios que fallan o entran en un estado no deseado.

En sistemas con systemd, la inicialización de scripts de configuración de daemons se realiza mediante unidades de servicio (unit files) que definen cómo systemd debe gestionar los servicios. Esto proporciona un enfoque más moderno y flexible para la gestión de servicios en sistemas con Linux, con características avanzadas como paralelización, reinicio automático y supervisión continua del estado de los servicios. Los comandos systemctl se utilizan para administrar los servicios y realizar operaciones como inicio, detención, reinicio, habilitación y deshabilitación.

Procesos

Un proceso en un sistema operativo es una instancia en ejecución de un programa. Cuando se ejecuta un programa en un sistema, el sistema operativo crea un proceso para él. Cada proceso tiene su propio espacio de memoria asignado y recursos asociados, como identificadores de proceso (PID), archivos abiertos, estado de la CPU, etc. De forma más detallada tenemos:

- Programa: Es el código ejecutable que se carga en la memoria para ser ejecutado.
- Espacio de memoria: Cada proceso tiene su propio espacio de memoria asignado que incluye código, datos y pila.
- Identificador de proceso (PID): Es un número único asignado por el sistema operativo para identificar un proceso.
- Contexto de ejecución: Contiene información sobre el estado actual del proceso, incluyendo el contador de programa, registros de la CPU, etc.
- Recursos: Los procesos pueden tener recursos asignados como archivos abiertos, conexiones de red, etc.

Podemos ver los procesos que están corriendo en este momento con el comando **ps**. Cuando lo corremos, vamos a ver una tabla donde cada fila es un proceso y tendremos las columnas:

PID	**TTY**	**TIME**	**CMD**
Process ID	Terminal en la que corre	Hace cuando se está ejecutando	Nombre del proceso o comando ejecutado

Cómo eliminar procesos

El comando **kill** se utiliza para enviar una señal a un proceso, lo que generalmente resulta en la terminación del proceso. La señal más comúnmente utilizada es **SIGTERM (15)**, que indica al proceso que debe terminar de manera limpia. Si el proceso no responde a **SIGTERM**, podemos enviar la señal **SIGKILL (9)** para forzar su terminación inmediata.

La sintaxis general de este comando es **kill [opción] [pid]** donde pid es el ID del proceso que queremos terminar y la opción es el tipo

de señal que queremos enviar al proceso. Algunas de estas señales incluyen:

- **SIGTERM (15):** Señal de terminación. Le indica al proceso que debe terminar de manera limpia. Es la señal predeterminada enviada por kill si no se especifica otra señal.
- **SIGKILL (9):** Señal de terminación forzada. Termina el proceso inmediatamente sin darle la oportunidad de realizar ningún tipo de limpieza o cierre adecuado. Esta señal es útil para detener procesos obstinados o que no responden.
- **SIGHUP (1):** Señal de colgar. Inicialmente diseñada para indicar a un proceso que debe volver a leer su configuración o reiniciarse. Algunos programas también utilizan esta señal para reiniciar o detener procesos.
- **SIGINT (2):** Señal de interrupción. Se utiliza comúnmente para interrumpir la ejecución de un proceso en el terminal, por ejemplo, presionando Ctrl + C. Algunos programas pueden tener un comportamiento específico cuando reciben esta señal.
- **SIGQUIT (3):** Señal de salida. Similar a SIGINT, pero suele ser utilizada para producir un volcado de núcleo (core dump) del proceso.
- **SIGSTOP (19):** Señal de detención. Detiene el proceso, pero no lo termina. El proceso puede ser reanudado posteriormente con SIGCONT.
- **SIGCONT (18):** Señal de continuación. Se utiliza para reanudar la ejecución de un proceso detenido previamente con SIGSTOP.

Algunos ejemplos para este comando son:

- Eliminar el proceso 1234 con SIGTERM: **kill 1234**.
- Forzar la terminación inmediata del proceso 1234 con SIGKILL: **kill -9 1234**.
- Eliminar varios procesos al mismo tiempo: **kill 1234 4321 4444**.
- Eliminar un proceso por nombre con pkill: **pkill apache2**.

Pipelines

Los pipelines, también conocidos como tuberías en español, son una característica fundamental en sistemas Linux que permiten la comunicación entre procesos de forma secuencial y eficiente. En un pipeline, la salida estándar (stdout) de un proceso se conecta directamente a la entrada estándar (stdin) de otro proceso, lo que permite que los datos fluyan de un proceso a otro de manera fluida. Esto facilita la creación de cadenas de procesos para realizar tareas más complejas de procesamiento de datos.

Algunas características clave de los pipelines incluyen:

- **Secuencialidad:** Los datos fluyen de un proceso a otro de manera secuencial, uno tras otro, en el orden en que se invocan los comandos en el pipeline.
- **Comunicación entre procesos:** Los pipelines permiten que los procesos se comuniquen entre sí de forma eficiente, sin necesidad de utilizar archivos temporales o mecanismos de comunicación más complejos.
- **Eficiencia:** Los pipelines son eficientes en términos de consumo de recursos, ya que los datos se procesan de forma incremental a medida que fluyen a través de los diferentes procesos en el pipeline.

La sintaxis básica de un pipeline en Linux es utilizar el operador | para conectar los comandos en el pipeline. Por ejemplo: comando1 | comando2 | comando3.

Ejemplos de pipelines:

- Contar las líneas de un archivo y ordenar el resultado alfabéticamente: **cat archivo.txt | wc -l | sort**. En este ejemplo, el comando cat muestra el contenido del archivo archivo.txt, la salida se pasa al comando wc -l que cuenta las líneas y luego la salida de wc -l se pasa al comando sort para ordenar el resultado alfabéticamente.

- Buscar archivos y contar cuántos hay: **find . -type f | wc -l**. En este ejemplo, el comando find busca archivos en el directorio actual (.) y sus subdirectorios. La lista de archivos encontrados se pasa a wc -l que cuenta la cantidad de líneas, que es equivalente a la cantidad de archivos encontrados.

- Filtrar líneas que contienen una palabra específica: **cat archivo.txt | grep "palabra"**. Este comando lee el contenido del archivo archivo.txt y filtra las líneas que contienen la palabra "palabra".

Redireccionamiento

Los caracteres **>** y **>>** son operadores de redirección en sistemas Linux que se utilizan para redirigir la salida de un comando hacia un archivo en lugar de hacia la salida estándar (generalmente la pantalla). Veamos un poco para que podemos usar cada uno.

Redirección de salida estándar

El operador > se utiliza para redirigir la salida estándar de un comando hacia un archivo. Si el archivo especificado no existe, se crea; si existe, se sobrescribe. Su sintaxis es **comando > archivo**.

Por ejemplo: ls > lista_archivos.txt. Este comando ejecuta ls para listar los archivos en el directorio actual y redirige la salida a un archivo llamado lista_archivos.txt. Si lista_archivos.txt ya existe, se sobrescribe con la nueva salida de ls.

Redirección de salida estándar con apéndice

El operador >> también se utiliza para redirigir la salida estándar de un comando hacia un archivo, pero en lugar de sobrescribir el archivo existente, agrega la salida al final del archivo. Su sintaxis es: **comando >> archivo**.

Por ejemplo: date >> registro.txt. Este comando ejecuta date para mostrar la fecha y la hora actuales, y agrega esta salida al final del archivo registro.txt. Si registro.txt no existe, se crea.

Hemos llegado así al final de esta clase en la que vimos:

- Administrar servicios en sistemas con SysV y systemd.
- Cómo ver y eliminar procesos.
- Cómo usar pipelines.
- Cómo redireccionar la salida de comandos.

Entender cómo funcionan estos sistemas nos ayudará a administrar apropiadamente servicios que queramos para el sistema operativo. Además, conocer cómo funcionan los pipelines y cómo redireccionar salidas de comandos nos va a permitir escribir comandos más complejos. La próxima clase haremos un repaso completo de lo visto hasta ahora

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Eckert, J. W. (2020). Linux+ and LPIC-1: Guide to Linux Certification. Cengage.

Para ampliar la información

IONOS. (s.f.). Comando ps de Linux. Recuperado de https://www.ionos.es/digitalguide/servidores/configuracion/comando-ps-de-linux/

Wikipedia. (s.f.). systemd. Recuperado de https://wiki.archlinux.org/title/systemd_(Espa%C3%B1ol)