Tema 2

Administración del kernel y procesos

Administración de Sistemas Operativos

Índice

[Esquema 3](#_Toc169858509)

[Material de estudio 4](#_Toc169858510)

[1.1. Introducción y objetivos 4](#_Toc169858511)

[1.2. Conceptos básicos del kernel 5](#_Toc169858512)

[1.3. Gestión de la memoria 6](#_Toc169858513)

[1.4. Gestión de procesos. 10](#_Toc169858514)

[1.4.1 Creación de procesos. 11](#_Toc169858515)

[1.4.2 Planificación de procesos. 11](#_Toc169858516)

[1.4.3 Estados de los procesos. 12](#_Toc169858517)

[1.4.4 Señales y comunicación entre procesos. 12](#_Toc169858518)

[1.5. Gestión de dispositivos. 13](#_Toc169858519)

[1.5.1 Gestión de dispositivos por el kernel. 13](#_Toc169858520)

[1.6 Gestión del sistema de archivos. 15](#_Toc169858521)

[1.7. Seguridad y control de acceso. 16](#_Toc169858522)

[1.8. Proceso de arranque del sistema Linux. 17](#_Toc169858523)

[1.8.1 BIOS/UEFI. 17](#_Toc169858524)

[1.8.2 Cargador de arranque (Bootloader). 18](#_Toc169858525)

[1.8.3 Inicialización del kernel. 18](#_Toc169858526)

[1.8.4 Sistema de inicialización (Init System). 18](#_Toc169858527)

[1.8.5 Systemd y demonios de inicio automático. 18](#_Toc169858528)

[1.4. Referencias bibliográficas 20](#_Toc169858529)

[A fondo 21](#_Toc169858530)

[Entrenamientos 22](#_Toc169858531)

[Test 27](#_Toc169858532)

Esquema

Tabla

Descripción generada automáticamente

Material de estudio

1.1. Introducción y objetivos

La administración del kernel y los procesos en Linux son algo fundamental para cualquier administrador de sistemas Linux. El kernel de Linux es el núcleo del sistema operativo, responsable de gestionar los recursos del hardware y proporcionar servicios esenciales a todas las aplicaciones del sistema. La correcta gestión del kernel y los procesos es crucial para garantizar un rendimiento óptimo, así como para la estabilidad y seguridad del sistema.

En este tema, se introducirán los conceptos esenciales y las prácticas avanzadas relacionadas con la administración del kernel y los procesos en un entorno Linux. Analizaremos cómo el kernel interactúa con el hardware, cómo se gestionan los procesos y los hilos de ejecución, y cómo se controla el arranque y la parada del sistema. Además, discutiremos las herramientas y técnicas disponibles para monitorear y ajustar el comportamiento del sistema, asegurando que funcione de manera eficiente y segura.

Los objetivos que se pretenden conseguir al final de este tema son:

* **Comprender la arquitectura y el funcionamiento del kernel de Linux.** Configurar y actualizar el kernel así cómo gestionar módulos del kernel.
* **Gestionar hilos de ejecución**. Diferenciar entre procesos e hilos de ejecución y administrar hilos y procesos concurrente.
* **Control de procesos.** Monitorizar y controlar procesos en ejecución, así como la utilización de comandos para gestionar procesos.
* **Entender el proceso de arranque del sistema Linux.** Gestionar servicios y demonios que se inician automáticamente. Apagar y reiniciar el sistema de manera segura.
* **Configurar parámetros del kernel para optimizar el rendimiento del sistema.** Realizar compilaciones personalizadas del kernel e implementar técnicas de troubleshooting para resolver problemas relacionados con el kernel.

1.2. Conceptos básicos del kernel

El kernel de Linux fue creado por Linus Torvalds en 1991. La motivación de Torvalds para crear Linux surgió de su insatisfacción con el sistema operativo MINIX, que era un sistema educativo basado en Unix. Quería desarrollar un sistema que fuera libremente disponible y que aprovechara las características avanzadas de los procesadores 80386 de Intel, particularmente la gestión de memoria virtual.

El 5 de octubre de 1991, Linus Torvalds lanzó la primera versión oficial del kernel de Linux, y desde entonces, el desarrollo del kernel ha sido un esfuerzo colaborativo de la comunidad de software libre. La primera versión que Torvalds lanzó era capaz de ejecutar bash (un shell de Unix) y gcc (el compilador de GNU), pero no era muy funcional.

El modelo de desarrollo abierto de Linux permitió que una comunidad global de desarrolladores contribuyera al código. Esto no solo aceleró el desarrollo, sino que también mejoró la calidad del software, ya que más ojos revisaban el código y sugerían mejoras.

A lo largo de los años, Linux ha incorporado muchas características avanzadas, como soporte para múltiples arquitecturas de hardware, sistemas de archivos avanzados (como ext3, ext4, XFS, y Btrfs), capacidades de red robustas, y un enfoque en la seguridad.

Hoy en día, el kernel de Linux es la base de una amplia variedad de sistemas operativos, incluyendo distribuciones populares como Ubuntu, Fedora, CentOS, y Debian, así como sistemas operativos especializados en servidores, dispositivos móviles (Android), sistemas embebidos y supercomputadoras.

El desarrollo del kernel de Linux sigue siendo uno de los mayores proyectos de software colaborativo en el mundo, con miles de desarrolladores contribuyendo a su código. Cada versión nueva del kernel introduce mejoras, soporte para nuevo hardware y correcciones de seguridad, manteniendo a Linux a la vanguardia de la tecnología de sistemas operativos.

El kernel es el núcleo del sistema operativo, actuando como intermediario entre el hardware y el software de un sistema informático. Es el componente central que gestiona las operaciones básicas del sistema, permitiendo que el software interactúe de manera eficiente con el hardware. En esencia, el kernel controla todas las operaciones del sistema, desde la gestión de la memoria hasta la ejecución de aplicaciones y la comunicación con dispositivos externos.

Las funciones principales del kernel son las siguientes:

* **Gestión de la Memoria:** Asigna y libera memoria para las aplicaciones y el sistema operativo.
* **Gestión de Procesos:** Maneja la creación, programación y terminación de procesos.
* **Gestión de Dispositivos:** Controla la interacción con el hardware a través de controladores de dispositivo.
* **Sistema de Archivos:** Proporciona una estructura para almacenar y organizar datos en dispositivos de almacenamiento.
* **Seguridad y Control de Acceso:** Asegura que los usuarios y aplicaciones solo puedan acceder a los recursos autorizados.

1.3. Gestión de la memoria

La gestión de la memoria es una de las funciones más cruciales del kernel de Linux. Su objetivo principal es garantizar que cada proceso tenga suficiente memoria para ejecutarse y que el sistema utilice de manera eficiente los recursos de memoria disponibles. La gestión de la memoria en Linux abarca varias técnicas y conceptos, incluyendo la paginación, la segmentación, la memoria virtual y la administración de intercambio (swap).

El kernel de Linux utiliza un esquema de gestión de memoria virtual que permite a los procesos utilizar más memoria de la que está físicamente disponible en el sistema. Cada proceso tiene su propio espacio de direcciones virtuales, proporcionando aislamiento y seguridad entre procesos. Esto significa que un proceso no puede acceder a la memoria de otro proceso, lo que ayuda a prevenir errores y mejorar la seguridad.

Paginación

La paginación es una técnica fundamental en la gestión de memoria del kernel de Linux. Divide la memoria física y virtual en bloques de tamaño fijo llamados páginas. En Linux, el tamaño típico de una página es de 4 KB, aunque puede variar según la arquitectura.

El kernel mantiene una estructura de datos conocida como tabla de páginas que mapea las direcciones virtuales de un proceso a direcciones físicas en la memoria RAM. Para acelerar el acceso a estas tablas de páginas, el hardware utiliza una caché llamada TLB (Translation Lookaside Buffer), que almacena las traducciones recientes de direcciones virtuales a físicas.

Segmentación

Aunque menos común en sistemas modernos, la segmentación es otra técnica utilizada en la gestión de memoria. Divide la memoria en segmentos de tamaños variables, cada uno representando una región lógica del programa, como código, datos y pila. En sistemas x86, la segmentación se utiliza junto con la paginación para gestionar la memoria de manera eficiente.

Memoria Virtual

La memoria virtual permite a los procesos utilizar un espacio de direcciones que es mayor que la memoria física disponible. Esto se logra mediante el uso de disco duro como extensión de la RAM, a través del mecanismo de swapping. Cada proceso tiene su propio espacio de direcciones virtuales, aislado del espacio de direcciones de otros procesos. Cuando la memoria física está llena, el kernel puede mover páginas inactivas de la memoria RAM al espacio de intercambio en el disco duro, liberando RAM para otros procesos.

Gestión de la Memoria Física

La memoria física en Linux está dividida en zonas, como ZONE\_DMA, ZONE\_NORMAL y ZONE\_HIGHMEM, cada una utilizada para diferentes propósitos según la arquitectura y el hardware. El kernel utiliza un algoritmo de asignación conocido como Buddy System para gestionar la memoria libre. Este sistema divide la memoria en bloques de tamaño de potencia de dos y combina o divide bloques según sea necesario, permitiendo una gestión eficiente de la memoria.

Gestión de la Memoria del Usuario y del Kernel

El espacio de direcciones virtuales del usuario es gestionado por el kernel y está protegido de acceso directo por otros procesos. La memoria utilizada por el kernel, por otro lado, no está paginada y es accesible a través de una parte especial del espacio de direcciones virtuales. Esto asegura que el kernel pueda operar de manera eficiente sin las limitaciones impuestas por la paginación.

Caché y Buffer

El kernel utiliza una caché de páginas para almacenar en caché las páginas de datos leídas desde el disco, mejorando el rendimiento del sistema al reducir el tiempo de acceso a los datos frecuentemente utilizados. Similarmente, el buffer cache almacena en caché bloques de dispositivos de almacenamiento, optimizando el acceso a los datos en estos dispositivos.

Memoria Compartida y Mapeo de Memoria

La memoria compartida permite que múltiples procesos compartan un segmento de memoria, facilitando la comunicación entre ellos. El mapeo de memoria (mmap) permite a los procesos mapear archivos o dispositivos en su espacio de direcciones virtuales, lo que permite el acceso a los contenidos de estos como si fueran parte de la memoria. Esto puede ser utilizado para optimizar el acceso a grandes volúmenes de datos y para la comunicación entre procesos.

Estructuras y Herramientas del Kernel para la Gestión de Memoria

El kernel utiliza varias estructuras de datos para gestionar la memoria. Por ejemplo, mm\_struct representa la información de gestión de memoria de un proceso, vm\_area\_struct representa un área contigua de memoria virtual dentro de un proceso, y page representa una página física en la memoria.

Varias herramientas y comandos están disponibles para ayudar a los administradores a monitorear y gestionar la memoria del sistema. El comando free muestra la cantidad de memoria libre y utilizada en el sistema, mientras que vmstat proporciona estadísticas del sistema relacionadas con la memoria, procesos y entrada/salida. Las herramientas top y htop muestran el uso de memoria por proceso en tiempo real, y smem analiza el uso de memoria de los procesos de manera más detallada, incluyendo la memoria compartida. Además, el archivo /proc/meminfo en el sistema de archivos proc contiene información detallada sobre el uso de la memoria del sistema.

Optimización y Problemas Comunes

La optimización de la memoria puede lograrse ajustando los parámetros del kernel a través de /proc/sys/vm/. Por ejemplo, el parámetro swappiness define la tendencia del kernel a usar swap. Herramientas como sysctl permiten modificar estos parámetros en tiempo de ejecución.

Algunos problemas comunes en la gestión de memoria incluyen la fragmentación de memoria, que puede causar que el sistema tenga suficientes bloques de memoria libres pero no contiguos, dificultando la asignación de grandes bloques de memoria. Otro problema es la memoria insuficiente (OOM - Out Of Memory), que ocurre cuando el sistema se queda sin memoria física y swap. En estos casos, el kernel puede invocar el OOM killer para terminar procesos y liberar memoria.

En resumen, la gestión de la memoria en Linux es un proceso complejo y sofisticado que involucra múltiples técnicas y estructuras para asegurar que los recursos de memoria sean utilizados eficientemente y que los procesos puedan ejecutarse de manera segura y efectiva. La comprensión y gestión adecuada de estos aspectos son cruciales para mantener el rendimiento y la estabilidad de un sistema Linux.

1.4. Gestión de procesos.

La gestión de procesos es una de las funciones más fundamentales y críticas del kernel de Linux. El kernel se encarga de crear, planificar, y finalizar procesos, asegurando que los recursos del sistema se utilicen de manera eficiente y que los procesos se ejecuten de manera segura y equitativa.

Conceptos Básicos

* **Proceso:** Un proceso es una instancia de un programa en ejecución. Cada proceso tiene su propio espacio de direcciones en memoria y sus propios recursos asignados.
* **PID (Process ID):** Cada proceso en el sistema tiene un identificador único llamado PID.
* **PPID (Parent Process ID):** El PID del proceso padre que creó el proceso actual.

Linux ofrece varios comandos útiles para la gestión de procesos:

* ps: muestra una lista de procesos en ejecución.
* top: proporciona una vista en tiempo real de los procesos en ejecución y su uso de recursos.
* htop: es similar a top, pero con una interfaz más amigable e interactiva.
* kill: se usa para enviar señales a los procesos, comúnmente para terminar procesos.
* nice y renice: permiten ajustar la prioridad de los procesos, lo que influye en cuánto tiempo de CPU reciben.

1.4.1 Creación de procesos.

La creación de procesos en Linux se realiza principalmente mediante dos llamadas al sistema: fork() y exec(). La llamada fork() se utiliza para duplicar el proceso actual, creando un nuevo proceso hijo con su propio espacio de direcciones y PID. Después de fork(), ambos procesos (padre e hijo) continúan ejecutándose desde el mismo punto. La llamada exec() se utiliza para reemplazar el espacio de direcciones del proceso actual con un nuevo programa, lo que es útil después de un fork() para ejecutar un programa diferente en el proceso hijo.

1.4.2 Planificación de procesos.

El kernel de Linux utiliza el planificador Completely Fair Scheduler (CFS) para decidir qué procesos deben ejecutarse y en qué orden. El objetivo del CFS es asignar tiempo de CPU de manera justa entre todos los procesos. Además, los procesos pueden tener prioridades que afectan cuánto tiempo de CPU reciben. Estas prioridades se pueden ajustar utilizando los comandos nice y renice, lo que permite a los usuarios aumentar o disminuir la prioridad de un proceso en particular.

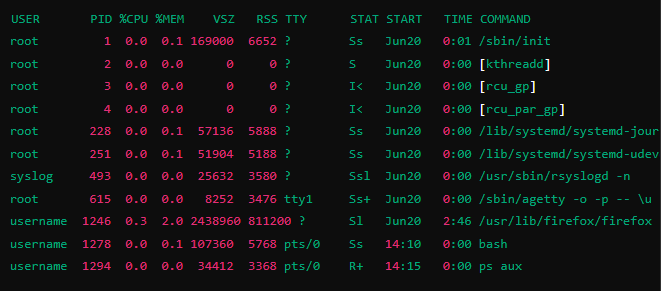
1.4.3 Estados de los procesos.

Los procesos en Linux pueden estar en varios estados, cada uno representando una etapa diferente en su ciclo de vida. Los estados principales son:

* **Running (R):** El proceso está ejecutándose o listo para ejecutarse.
* **Sleeping (S):** El proceso está esperando un evento, como la finalización de una operación de entrada/salida.
* **Stopped (T):** El proceso ha sido detenido, generalmente por una señal.
* **Zombie (Z):** El proceso ha terminado, pero su entrada en la tabla de procesos aún no ha sido limpiada por su proceso padre.

A continuación se muestra un ejemplo de uso del comando ps para mostrar una lista de los procesos actuales en el sistema:





1.4.4 Señales y comunicación entre procesos.

Las señales son una forma de comunicación entre procesos en Linux. Permiten que los procesos se notifiquen entre sí sobre eventos o que soliciten acciones específicas. Por ejemplo, una señal común es SIGTERM, que pide a un proceso que se termine. Los usuarios pueden enviar señales a los procesos utilizando el comando kill. Cada tipo de señal tiene un propósito específico, y los procesos pueden definir manejadores de señales para realizar acciones específicas cuando reciben una señal.

1.5. Gestión de dispositivos.

El kernel de Linux gestiona los dispositivos de hardware mediante controladores de dispositivo (device drivers). Estos controladores actúan como intermediarios entre el hardware y el software del sistema operativo, permitiendo que el kernel y las aplicaciones interactúen con el hardware de manera abstracta y segura.

Los controladores de dispositivo son programas que permiten al kernel interactuar con dispositivos de hardware específicos. Cada tipo de dispositivo, como discos duros, tarjetas de red y teclados, tiene su propio controlador. Los controladores operan en el espacio del kernel, donde tienen acceso completo al hardware y a la memoria del sistema, a diferencia de las aplicaciones que operan en el espacio de usuario.

1.5.1 Gestión de dispositivos por el kernel.

Durante el arranque, el kernel de Linux detecta todos los dispositivos de hardware presentes en el sistema utilizando mecanismos como ACPI y PnP. Luego, el kernel carga los controladores necesarios, que pueden estar integrados en el kernel o ser módulos que se cargan dinámicamente. El subsistema udev gestiona dinámicamente los dispositivos, cargando los módulos necesarios cuando se detecta un nuevo dispositivo. Los controladores proporcionan una interfaz estándar que el kernel y las aplicaciones pueden usar para interactuar con el hardware, ocultando los detalles específicos del hardware y proporcionando una capa de abstracción.

Existen varios tipos de controladores:

* **Controladores de Caracteres:** Gestionan dispositivos que envían y reciben datos en un flujo continuo, como teclados y puertos serie.
* **Controladores de Bloques:** Gestionan dispositivos que almacenan datos en bloques, como discos duros y SSDs.
* **Controladores de Red:** Gestionan dispositivos de red, como tarjetas de red Ethernet y adaptadores WiFi.

Los dispositivos de hardware se representan como archivos especiales en el directorio /dev. Las aplicaciones pueden leer y escribir en estos archivos para interactuar con los dispositivos. Utilidades de línea de comandos como lsmod, modprobe, lspci, y lsusb proporcionan información y control sobre los dispositivos y sus controladores.

* **lsmod:** Lista los módulos del kernel cargados.
* **modprobe:** Carga o descarga módulos del kernel.
* **lspci y lsusb:** Muestran información sobre dispositivos PCI y USB, respectivamente.
* **udevadm:** Inspecciona y controla el estado de los dispositivos gestionados por udev.

Para ver las particiones de un disco, se puede utilizar fdisk o lsblk.



Para montar un sistema de archivos, se usa el comando mount, y para desmontarlo, se usa umount.





1.6 Gestión del sistema de archivos.

El kernel de Linux proporciona una estructura para almacenar y organizar datos en dispositivos de almacenamiento mediante el uso de sistemas de archivos. Un sistema de archivos es un método que el sistema operativo utiliza para controlar cómo se almacenan y recuperan los datos.

Cuando se conecta un dispositivo de almacenamiento (como un disco duro o una memoria USB) al sistema, el kernel detecta el dispositivo y carga los controladores necesarios para interactuar con él. Luego, el kernel puede montar el sistema de archivos del dispositivo, lo que hace que su contenido sea accesible para el usuario y las aplicaciones.

El proceso de organización y almacenamiento de datos incluye varias capas y componentes:

* **Bloques de Datos:** El dispositivo de almacenamiento se divide en bloques de datos, que son las unidades básicas de almacenamiento. El tamaño de estos bloques puede variar, pero comúnmente es de 4KB.
* **Superbloque:** Contiene información esencial sobre el sistema de archivos, como su tamaño, el número de bloques y otros parámetros importantes.
* **Inodos:** Cada archivo y directorio tiene un inodo, que es una estructura de datos que almacena información sobre el archivo, como su tamaño, permisos, y ubicaciones de los bloques de datos.
* **Directorios:** Los directorios son listas de inodos y nombres de archivos. Permiten organizar los archivos en una estructura jerárquica, similar a un árbol, donde los directorios pueden contener otros directorios y archivos.
* **Gestión del Espacio Libre:** El kernel mantiene una lista de bloques de datos libres que pueden ser asignados a nuevos archivos. Esto asegura que el espacio en el dispositivo de almacenamiento se utilice de manera eficiente.

Para hacer que los archivos y directorios de un dispositivo de almacenamiento sean accesibles, el sistema de archivos se "monta" en un punto de montaje del sistema de archivos del sistema operativo. Por ejemplo, cuando montas una memoria USB, puedes acceder a sus archivos a través de un directorio como /mnt/usb.

Además de estos componentes básicos, el kernel de Linux soporta múltiples tipos de sistemas de archivos, cada uno optimizado para diferentes tipos de uso. Algunos ejemplos de sistemas de archivos comunes en Linux son ext4, XFS y Btrfs. Cada uno de estos sistemas de archivos tiene sus propias características y métodos de gestión de datos, pero todos siguen los principios básicos de bloques de datos, inodos y directorios para organizar y almacenar datos de manera eficiente.

En resumen, el kernel de Linux proporciona una estructura robusta para almacenar y organizar datos en dispositivos de almacenamiento mediante el uso de sistemas de archivos, que gestionan la asignación de bloques, mantienen información sobre archivos y directorios, y aseguran un acceso eficiente y seguro a los datos.

1.7. Seguridad y control de acceso.

El kernel de Linux asegura que los usuarios y aplicaciones solo puedan acceder a los recursos autorizados mediante un sistema de permisos y control de acceso. Este sistema garantiza la seguridad y la integridad de los datos y recursos del sistema. A continuación se explica cómo funciona de manera clara y sencilla.

Primero, cada archivo y directorio en Linux tiene asociados permisos de acceso que definen qué usuarios pueden leer, escribir o ejecutar ese archivo o directorio. Estos permisos están divididos en tres categorías: propietario (user), grupo (group) y otros (others). Para cada categoría, se pueden establecer permisos de lectura (r), escritura (w) y ejecución (x).

Por ejemplo, un archivo con permisos rwxr-xr-- significa que el propietario puede leer, escribir y ejecutar el archivo, los miembros del grupo pueden leer y ejecutar el archivo, y los demás usuarios solo pueden leer el archivo.

Además de los permisos de archivos, el kernel utiliza identificadores de usuario (UID) y de grupo (GID) para gestionar el acceso. Cada usuario en el sistema tiene un UID y cada grupo tiene un GID. Cuando un usuario intenta acceder a un archivo o recurso, el kernel compara su UID y GID con los permisos del archivo para determinar si el acceso está permitido.

El kernel también implementa el concepto de procesos y sus permisos asociados. Cada proceso tiene un UID y GID que determinan sus permisos. Un proceso solo puede acceder a los archivos y recursos para los que su UID y GID tienen permisos suficientes.

Para mejorar aún más la seguridad, Linux utiliza el modelo de capacidades, que permite restringir las acciones que pueden realizar incluso los usuarios con privilegios de superusuario (root). Por ejemplo, puedes configurar un proceso para que tenga permisos de administración de red pero no permisos para modificar archivos del sistema.

1.8. Proceso de arranque del sistema Linux.

El proceso de arranque de un sistema Linux empieza desde el momento en que se enciende la computadora hasta que el sistema operativo está completamente listo para su uso.

1.8.1 BIOS/UEFI.

Cuando se enciende la computadora, lo primero que se ejecuta es el BIOS (Basic Input/Output System) o UEFI (Unified Extensible Firmware Interface). Su trabajo es inicializar y probar el hardware de la computadora, como la memoria y los dispositivos de entrada/salida (teclado, pantalla, etc.). Luego, busca un dispositivo de arranque, como un disco duro o una memoria USB, donde se encuentra el sistema operativo.

1.8.2 Cargador de arranque (Bootloader).

Una vez que el BIOS/UEFI encuentra un dispositivo de arranque, carga el bootloader en la memoria. El bootloader es un pequeño programa que se encarga de cargar el sistema operativo. En Linux, el bootloader más común es GRUB (GRand Unified Bootloader). GRUB muestra un menú donde puedes elegir qué sistema operativo o kernel de Linux quieres iniciar. Luego, carga el kernel seleccionado en la memoria y transfiere el control a este.

1.8.3 Inicialización del kernel.

El kernel es el núcleo del sistema operativo. Una vez cargado, el kernel se encarga de detectar y configurar el hardware de la computadora. También monta el sistema de archivos raíz, lo que significa que prepara el sistema para poder acceder a los archivos y directorios necesarios. Después de esta configuración inicial, el kernel ejecuta el primer proceso en el espacio de usuario, que es el proceso de inicialización (init).

1.8.4 Sistema de inicialización (Init System).

El proceso de inicialización varía según el sistema de init utilizado. Los sistemas de init más comunes en Linux son systemd, SysVinit, y Upstart. Systemd es el más usado en las distribuciones modernas de Linux, como Ubuntu, Fedora y Debian. El sistema de init se encarga de arrancar todos los servicios necesarios para que el sistema esté listo para su uso.

1.8.5 Systemd y demonios de inicio automático.

systemd es un sistema de inicialización y gestor de servicios que paraleliza el inicio de servicios, lo que mejora la capacidad de respuesta del sistema. Los servicios, también llamados demonios, son programas que se ejecutan en segundo plano para realizar tareas esenciales.

* **Configuración de systemd:** Los archivos de configuración de systemd se encuentran en /etc/systemd/system/ y /lib/systemd/system/. Cada servicio tiene un archivo de unidad (unit file) que define cómo y cuándo debe iniciarse.
* **Inicialización de Servicios:** systemd lee los archivos de unidad y arranca los servicios necesarios de acuerdo con sus dependencias y el objetivo (target) predeterminado, como multi-user.target para un sistema sin interfaz gráfica o graphical.target para un sistema con entorno gráfico.
* **Servicios Comunes de Inicio Automático:**
  + systemd-journald: Gestiona y almacena los registros del sistema (logs).
  + systemd-udevd: Gestiona los eventos de dispositivos (plug and play).
  + NetworkManager: Gestiona las conexiones de red.
  + cron: Gestiona tareas programadas (cron jobs).
  + sshd: Proporciona acceso remoto seguro mediante SSH.
  + cups: Gestiona la impresión.
  + rsyslog: Gestiona los registros del sistema si se utiliza rsyslog para logging.

systemd utiliza el comando systemctl para gestionar los servicios:

* **Para iniciar un servicio:**



* **Para habilitar un servicio para que se inicie automáticamente en el arranque:**



* **Para verificar el estado de un servicio:**



1.9. Referencias bibliográficas

Caballero, A. J., Administración de Sistemas Operativos. Ra-Ma, 2017.

García Carrasco, J. (2016). Administración de Sistemas Operativos. Editorial Síntesis.

Pons, N., Linux. Principios básicos de uso del sistema. Ediciones ENI, 2023.

Rohaut, S., Linux. Dominar la administración del sistema. Ediciones ENI, 2021.

A fondo

Comando Chmod: Cómo cambiar permisos de archivo en Linux.

Rosa, D. (2024). *Comando Chmod: Cómo cambiar permisos de archivo en Linux.* [FreeCodeCamp.](https://www.redalyc.org/articulo.oa?id=337630234004) https://www.freecodecamp.org/espanol/news/comando-chmod-como-cambiar-permisos-de-archivo-en-linux/

Esta página web explica al detalle los permisos que pueden existir, cómo leerlos y los pasos a seguir para su modificación.

Manual de instalación de Ubuntu sobre VirutalBox

ProgrammingKnowledge. (2024). *How to install Ubuntu 24,04 TLS on VirtualBox in Windows 11* [Vídeo]. Youtube. https://youtu.be/DhVjgI57Ino?si=csk63pm3nlYZa2jW

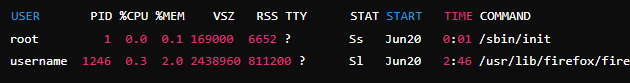
En este vídeo te muestra cómo instalar la herramienta VirtualBox como contenedor de máquinas virtuales y cómo descargar y añadir un sistema operativo Ubuntu.

Entrenamientos

Entrenamiento 1

* Exploración de los procesos del sistema. Aprende a visualizar y entender los procesos que se están ejecutando en el sistema.
* Paso a paso:
  + Abre una terminal en tu sistema Linux.
  + Usa el comando ps para listar todos los procesos que se están ejecutando: La salida mostrará una lista de procesos con información como el PID, usuario, %CPU, %MEM, y el comando.
  + Identifica los siguientes detalles para algunos procesos:
    - PID (Process ID)
    - Usuario que está ejecutando el proceso
    - Porcentaje de uso de CPU y memoria
    - Comando que inició el proceso
  + Usa el comando top para monitorear los procesos en tiempo real: Observa la lista de procesos en tiempo real, ordenada por el uso de CPU.
    - El comando touch crea un nuevo archivo bajo tu ruta actual
    - Chmod cambia todos los permisos al usuario, permisos de lectura y escritura al grupo, y ningún permiso al resto
* Solución

ps aux



top

Entrenamiento 2

* Manipulación de prioridades de procesos. Aprende a cambiar la prioridad de los procesos usando nice y renice.
* Paso a paso:
  + Abre una terminal y ejecuta un proceso que consuma tiempo de CPU, por ejemplo, el proceso yes.
  + Usa el comando ps para encontrar el PID de este proceso
  + Ajusta la prioridad del proceso usando renice para aumentar y disminuir su prioridad.
  + Observa cómo cambia el comportamiento del proceso usando el comando top: El cambio de prioridad debería reflejarse en la lista de top, con el proceso yes moviéndose hacia arriba o abajo en la lista según la prioridad.
  + Detén el proceso una vez que hayas terminado: El proceso yes será terminado.
* Solución

yes > /dev/null &

ps aux | grep yes



sudo renice 10 -p <PID>

sudo renice -10 -p <PID>

top

kill <PID>

Entrenamiento 3

* Configuración y uso de sysctl. Aprende a ajustar los parámetros del kernel para optimizar el rendimiento del sistema.
* Paso a paso:
  + Abre una terminal y visualiza los valores actuales de los parámetros del kernel usando sysctl: Se listarán todos los parámetros del kernel y sus valores actuales.
  + Ajusta el parámetro de swappiness para reducir el uso de swap: Establece el valor de swappiness a 10 temporalmente.
  + Comprueba que el cambio se ha aplicado correctamente: Debería mostrar vm.swappiness = 10.
  + Haz que el cambio sea permanente editando el archivo /etc/sysctl.conf añadiendo la siguiente línea al archivo: vm.swappiness=10
* Solución

sudo sysctl -a

sudo sysctl vm.swappiness=10

sudo sysctl vm.swappiness

sudo nano /etc/sysctl.conf

vm.swappiness=10

Entrenamiento 4

* Compilación de un Kernel personalizado. Aprende los pasos básicos para compilar un kernel personalizado.
* Paso a paso:
  + Descarga el código fuente del kernel desde [kernel.org](https://www.kernel.org)
  + Copia la configuración actual del kernel.
  + Configura el kernel usando make menuconfig: Navega por las opciones de configuración, pero no realices cambios importantes.
  + Compila el kernel y los módulos.
  + Instala los módulos y el kernel.
  + Actualiza GRUB y reinicia el sistema.
* Solución

wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.10.tar.xz

tar -xvf linux-5.10.tar.xz

cd linux-5.10

cp /boot/config-$(uname -r) .config

make menuconfig

make -j$(nproc)

make modules

sudo make modules\_install

sudo make install

sudo update-grub

sudo reboot

Entrenamiento 5

* Monitorización y solución de problemas del sistema. Aprender a usar herramientas básicas de troubleshooting para resolver problemas del kernel.
* Paso a paso:
  + Abre una terminal y usa dmesg para revisar los mensajes del kernel: Revisa los mensajes de registro para identificar cualquier error o advertencia.
  + Identifica cualquier mensaje de error o advertencia en los registros del sistema: Busca mensajes destacados que indiquen problemas, como fallos de hardware o problemas con controladores.
  + Usa strace para rastrear el comportamiento de un comando, por ejemplo ls.
  + Usa lsmod para listar los módulos del kernel cargados y modprobe para cargar o descargar un módulo.
* Solución

dmesg | less

strace ls

lsmod

sudo modprobe -r <módulo>

sudo modprobe <módulo>

Test

1. ¿Cuál es el propósito principal del comando ps en Linux?:

A) Reiniciar el sistema

\_B) Mostrar una lista de procesos en ejecución

C) Configurar los parámetros del kernel

D) Compilar el kernel.

El comando ps muestra una instantánea de los procesos actuales en el sistema, proporcionando información sobre el PID, usuario, uso de CPU, memoria y el comando que inició el proceso.

1. ¿Qué archivo contiene la configuración actual del kernel que se puede usar para compilar un nuevo kernel?

A) /etc/fstab

B) /boot/grub/grub.cfg

C) /etc/sysctl.conf

\_D) /boot/config-$(uname -r)

El archivo /boot/config-$(uname -r) contiene la configuración actual del kernel en uso, que puede ser utilizada como base para compilar un nuevo kernel.

1. ¿Cuál es el sistema de init más común en las distribuciones modernas de Linux como Ubuntu y Fedora?

A) SysVinit

B) Upstart

\_C) systemd

D) init

systemd es el sistema de inicialización y gestión de servicios más común en las distribuciones modernas de Linux, reemplazando a SysVinit y Upstart.

1. ¿Qué herramienta se utiliza para ajustar los parámetros del kernel en tiempo real?

A) fdisk

\_B) sysctl

C) top

D) make

sysctl es una herramienta que permite ver y cambiar los parámetros del kernel en tiempo real.

1. ¿Cuál es el comando para mostrar los mensajes del kernel en Linux?

A) top

B) ps

\_C) dmesg

D) lsmod

dmesg muestra los mensajes del kernel, incluyendo información sobre el arranque del sistema y eventos de hardware.

1. ¿Qué comando se usa para cargar un módulo del kernel?

A) lsmod

\_B) modprobe

C) insmod

D) rmmod

modprobe se utiliza para cargar módulos del kernel y sus dependencias automáticamente.

1. ¿Cuál de los siguientes comandos muestra una lista de módulos del kernel actualmente cargados?

A) top

B) ps

\_C) lsmod

D) sysctl

lsmod lista todos los módulos del kernel que están actualmente cargados en el sistema.

1. ¿Qué significa el parámetro vm.swappiness en el kernel de Linux?

A) Controla la frecuencia de las actualizaciones del sistema

\_B) Controla cuánto el kernel usará la memoria swap

C) Controla el tamaño del sistema de archivos

D) Controla la prioridad de los procesos

vm.swappiness determina cuánto el kernel prefiere usar la memoria swap en lugar de la RAM física. Un valor más bajo reduce el uso de swap.

1. ¿Cuál es el comando para cambiar la prioridad de un proceso en ejecución?

A) ps

\_B) renice

C) top

D) kill

renice se utiliza para cambiar la prioridad de un proceso en ejecución, permitiendo aumentar o disminuir su prioridad de CPU.

1. ¿Qué comando se usa para compilar el kernel de Linux después de configurarlo?

\_A) make

B) gcc

C) compile

D) build

make es el comando utilizado para compilar el kernel de Linux y sus módulos después de configurarlos.

1. ¿Cómo se llama el proceso que siempre tiene PID 1 en un sistema Linux?

A) kernel

B) systemd

\_C) init

D) bash

El proceso init (o systemd en sistemas modernos) siempre tiene PID 1 y es el primer proceso que se ejecuta en el espacio de usuario durante el arranque del sistema.

1. ¿Qué herramienta proporciona una vista en tiempo real de los procesos en ejecución y su uso de recursos?

A) ps

B) strace

C) sysctl

\_D) top

top proporciona una vista en tiempo real de los procesos en ejecución, mostrando el uso de CPU, memoria y otros recursos.

1. ¿Qué archivo se edita para hacer permanentes los cambios realizados con sysctl?

A) /etc/fstab

B) /boot/grub/grub.cfg

\_C) /etc/sysctl.conf

D) /etc/rc.local

Los cambios realizados con sysctl se pueden hacer permanentes añadiéndolos al archivo /etc/sysctl.conf.

1. ¿Cuál de los siguientes comandos se usa para monitorizar las llamadas al sistema realizadas por un proceso?

\_A) strace

B) sysctl

C) dmesg

D) top

strace se utiliza para rastrear y monitorizar las llamadas al sistema realizadas por un proceso.

1. ¿Qué comando se utiliza para iniciar un servicio administrado por systemd?

A) systemd start

B) sysctl start

C) service start

\_D) systemctl start

systemctl start <nombre\_del\_servicio> se utiliza para iniciar un servicio administrado por systemd.

1. ¿Cuál es el objetivo principal del comando nice en Linux?

A) Cambiar el nombre de un archivo

\_B) Cambiar la prioridad de ejecución de un proceso

C) Iniciar un nuevo proceso

D) Matar un proceso en ejecución

nice se utiliza para iniciar un proceso con una prioridad modificada, permitiendo ajustar cuánto tiempo de CPU recibe.

1. ¿Qué archivo contiene las configuraciones de GRUB en la mayoría de las distribuciones Linux?

A) /etc/fstab

\_B) /boot/grub/grub.cfg

C) /etc/sysctl.conf

D) /etc/init.d

/boot/grub/grub.cfg contiene las configuraciones de GRUB, el gestor de arranque utilizado en muchas distribuciones de Linux.

1. ¿Qué comando se usa para verificar y ajustar los parámetros del kernel de Linux?

A) lsmod

\_B) sysctl

C) strace

D) ps

sysctl se utiliza para ver y cambiar los parámetros del kernel en tiempo real.

1. ¿Qué hace el comando kill en Linux?

A) Detiene un servicio

\_B) Cierra un proceso en ejecución

C) Inicia un nuevo proceso

D) Cambia la prioridad de un proceso

kill envía una señal a un proceso para terminar su ejecución.

1. ¿Cuál es el propósito del archivo /etc/sysctl.conf?

\_A) Configurar los parámetros del kernel de manera permanente

B) Almacenar los logs del sistema

C) Configurar los servicios de inicio

D) Definir las reglas de firewall

/etc/sysctl.conf se utiliza para definir los parámetros del kernel que deben aplicarse de manera permanente al arrancar el sistema.

1. ¿Qué es el proceso yes en Linux?

A) Un editor de texto

B) Un comando para listar archivos

\_C) Un comando que genera una salida continua de una cadena de texto

D) Un gestor de paquetes

El proceso yes genera una salida continua de una cadena de texto (por defecto "y") hasta que se detenga manualmente.

1. ¿Cuál es el propósito principal de make menuconfig al compilar el kernel de Linux?

A) Ver los procesos en ejecución

\_B) Configurar opciones del kernel mediante una interfaz de menú

C) Instalar paquetes del kernel

D) Monitorizar el uso de la CPU

make menuconfig se utiliza para configurar las opciones del kernel mediante una interfaz de menú antes de compilarlo.