Introducción a los Sistemas Operativos

# Práctica 2

1. Editor de textos:

(a) Nombre al menos 3 editores de texto que puede utilizar desde la línea de comandos.

1. Vi: Es un editor de texto modal que se encuentra preinstalado en la mayoría de las distribuciones de Linux. Es muy potente y cuenta con una gran cantidad de funciones, pero también puede ser un poco complicado de aprender al principio.
2. Nano: Es un editor de texto más simple que Vi y es una buena opción para aquellos que no están familiarizados con Vi. Nano también se encuentra preinstalado en la mayoría de las distribuciones de Linux.
3. Emacs: Es un editor de texto muy potente y altamente personalizable. Es una buena opción para aquellos que necesitan un editor de texto con una gran cantidad de funciones. Emacs puede ser un poco más difícil de aprender que Vi o Nano, pero una vez que se domina, es un editor de texto muy eficiente.

(b) ¿En qué se diferencia un editor de texto de los comandos cat, more o less? Enumere los modos de operación que posee el editor de textos vi.

Un editor de texto, como Vim, Emacs o Nano, es un programa que se utiliza para crear, editar y guardar archivos de texto. En cambio, los comandos cat, more y less son programas que se utilizan para leer archivos de texto en la terminal.

En cuanto a los modos de operación del editor de texto vi, estos son los principales:

* Modo de comando: Es el modo por defecto en el que se encuentra vi. En este modo se pueden ejecutar comandos, moverse por el archivo, realizar búsquedas y otros tipos de operaciones.
* Modo de inserción: Este modo permite insertar texto en el archivo. Se ingresa al modo de inserción presionando la tecla "i" o cualquier otra tecla de inserción.
* Modo de reemplazo: Este modo permite reemplazar caracteres y texto. Se ingresa al modo de reemplazo presionando la tecla "R".
* Modo visual: Este modo permite seleccionar texto con el cursor y luego realizar operaciones con ese texto, como copiar, pegar o eliminar. Se ingresa al modo visual presionando la tecla "v".
* Modo ex: Este modo es para ejecutar comandos de ex, que es un editor de línea de comandos similar a vi. Se ingresa al modo ex presionando la tecla ":" en el modo de comando.

(c) Nombre los comandos más comunes que se le pueden enviar al editor de textos vi.

Algunos de los comandos más comunes que se pueden enviar al editor de textos Vi:

* Modo de comando:
  + :w -> Guarda el archivo.
  + :q -> Sale del editor.
  + :wq -> Guarda el archivo y sale del editor.
  + :q! -> Sale del editor sin guardar cambios.
  + yy -> Copia una línea de texto.
  + p -> Pega una línea de texto.
  + dd -> Elimina una línea de texto.
  + /texto -> Busca la cadena de texto hacia adelante.
  + ?texto -> Busca la cadena de texto hacia atrás.
  + n -> Busca la siguiente coincidencia.
  + N -> Busca la coincidencia anterior.
  + :set number -> Muestra el número de línea en la ventana de edición.
  + :set nonumber -> Oculta el número de línea en la ventana de edición.
* Modo de inserción:
  + i -> Ingresa al modo de inserción antes del cursor.
  + I -> Ingresa al modo de inserción al principio de la línea.
  + a -> Ingresa al modo de inserción después del cursor.
  + A -> Ingresa al modo de inserción al final de la línea.
  + -> Ingresa al modo de inserción en una nueva línea después de la línea actual.
  + -> Ingresa al modo de inserción en una nueva línea antes de la línea actual.

2. Proceso de Arranque SystemV :

(a) Enumere los pasos del proceso de inicio de un sistema GNU/Linux, desde que se prende la PC hasta que se logra obtener el login en el sistema.

El proceso de inicio de un sistema GNU/Linux que utiliza el sistema de inicio SystemV consta de varios pasos que se ejecutan en secuencia:

1. BIOS: La BIOS (Sistema Básico de Entrada/Salida) se encarga de realizar un autodiagnóstico del hardware y de encontrar un dispositivo de arranque, como un disco duro o un USB.
2. Carga del gestor de arranque: Si hay un gestor de arranque, como GRUB, se cargará en la memoria y se mostrará en la pantalla. El gestor de arranque permite al usuario seleccionar el sistema operativo que desea iniciar, entre otras opciones.
3. Carga del kernel: El gestor de arranque carga el kernel de Linux en la memoria y lo inicializa. El kernel es el corazón del sistema operativo y se encarga de administrar los recursos del hardware y del software.
4. Proceso de inicialización: El kernel inicia el proceso de inicialización y ejecuta los scripts de inicio ubicados en el directorio /etc/init.d/. Los scripts de inicio se encargan de iniciar los servicios del sistema y configurar el entorno de trabajo.
5. Ejecución de runlevels: El sistema de inicio SystemV utiliza runlevels, que son niveles de ejecución que definen los servicios que se ejecutan en el sistema. El kernel carga el runlevel predeterminado y ejecuta los scripts de inicio asociados con ese runlevel.
6. Arranque de servicios: Los scripts de inicio se encargan de arrancar los servicios del sistema, como el servidor de X, el servidor de red, el sistema de archivos, etc.
7. Inicio de sesión: Una vez que se han iniciado todos los servicios del sistema, se muestra la pantalla de inicio de sesión, donde el usuario debe ingresar su nombre de usuario y contraseña para acceder al sistema.
8. Shell de inicio de sesión: Una vez que se ha autenticado el usuario, se carga el shell de inicio de sesión, que es un programa que proporciona una interfaz de línea de comandos para el usuario. En un sistema GNU/Linux, el shell de inicio de sesión predeterminado es Bash.

(b) Proceso **INIT**. ¿Quién lo ejecuta? ¿Cuál es su objetivo?

El proceso INIT es ejecutado por el kernel de Linux como el primer proceso de usuario en el sistema, después de la carga del kernel y antes de cualquier otro proceso en el sistema. El objetivo principal del proceso INIT es iniciar y mantener en funcionamiento todos los procesos y servicios que son esenciales para el sistema operativo.

El proceso INIT es responsable de llevar a cabo la secuencia de inicio del sistema y de coordinar el inicio y finalización de los procesos y servicios. Además, el proceso INIT también es responsable de cambiar el runlevel del sistema en función de las necesidades del usuario.

En un sistema basado en SystemV, el proceso INIT maneja el cambio de runlevel del sistema. El runlevel es un modo de operación del sistema que determina qué servicios y procesos se ejecutan en el sistema. El proceso INIT utiliza los scripts de inicio y detención ubicados en el directorio /etc/init.d/ para iniciar y detener los servicios y procesos que se ejecutan en cada runlevel.

(c) Ejecute el comando pstree. ¿Qué es lo que se puede observar a partir de la ejecución de este comando?

El comando "pstree" muestra una representación en árbol de todos los procesos en ejecución en el sistema. Al ejecutar el comando "pstree" en una terminal, se puede observar la jerarquía de los procesos del sistema, comenzando por el proceso "init" y sus procesos hijos. Cada proceso hijo está representado como una rama del proceso padre en el árbol.

(d) RunLevels. ¿Qué son? ¿Cuál es su objetivo?

Los runlevels (niveles de ejecución) son un conjunto de modos de operación del sistema en GNU/Linux que definen qué servicios y procesos deben ser iniciados durante el proceso de arranque del sistema operativo.

En Linux, los runlevels se utilizan para controlar qué procesos y servicios se ejecutan en el sistema en diferentes situaciones, como el inicio del sistema, el reinicio del sistema, la recuperación del sistema después de una falla, entre otros. Cada runlevel se define por una combinación de servicios y procesos que se ejecutan en el sistema.

En general, los runlevels son numerados de 0 a 6, donde cada número representa un estado específico del sistema:

* 0: Apagado del sistema.
* 1: Modo de usuario único. Solo se ejecuta un número mínimo de procesos y servicios para permitir al administrador realizar tareas de mantenimiento.
* 2: Modo multiusuario sin red. El sistema se inicia con un número mínimo de servicios y procesos para permitir la conexión de un usuario local.
* 3: Modo multiusuario completo. El sistema se inicia con todos los servicios y procesos necesarios para admitir conexiones de red.
* 4: Sin uso definido por defecto, el administrador del sistema puede definirlo según sea necesario.
* 5: Modo multiusuario completo con entorno gráfico. El sistema se inicia con todos los servicios y procesos necesarios para admitir conexiones de red y un entorno gráfico de usuario.
* 6: Reinicio del sistema.

El objetivo principal de los runlevels es permitir al administrador del sistema controlar el comportamiento del sistema durante el proceso de arranque. Al elegir un runlevel determinado, el administrador puede definir qué servicios y procesos se ejecutan en el sistema, lo que permite ajustar el rendimiento, la seguridad y la funcionalidad del sistema operativo según sea necesario.

(e) ¿A qué hace referencia cada nivel de ejecución según el estándar? ¿Dónde se define qué Runlevel ejecutar al iniciar el sistema operativo? ¿Todas las distribuciones respetan estos estándares?

El estándar de los runlevels en GNU/Linux fue definido por la organización Linux Standard Base (LSB) y es ampliamente respetado por la mayoría de las distribuciones de GNU/Linux. Según este estándar, cada nivel de ejecución se define de la siguiente manera:

* Nivel de ejecución 0: El sistema se apaga.
* Nivel de ejecución 1: El sistema se inicia en modo de usuario único o modo de mantenimiento, en el que solo se ejecuta un número mínimo de servicios y procesos para permitir que el administrador realice tareas de mantenimiento en el sistema.
* Nivel de ejecución 2: El sistema se inicia en modo multiusuario sin red.
* Nivel de ejecución 3: El sistema se inicia en modo multiusuario completo, con todos los servicios y procesos necesarios para admitir conexiones de red.
* Nivel de ejecución 4: El nivel 4 se reserva para un uso futuro y su implementación depende de la distribución de GNU/Linux que se esté utilizando.
* Nivel de ejecución 5: El sistema se inicia en modo multiusuario completo con entorno gráfico.
* Nivel de ejecución 6: El sistema se reinicia.

El runlevel que se ejecuta al iniciar el sistema operativo se define en el archivo "/etc/inittab" en la mayoría de las distribuciones de GNU/Linux. En este archivo, se especifica el nivel de ejecución predeterminado mediante la línea "id:5:initdefault:", donde el número 5 representa el runlevel que se ejecutará por defecto.

No todas las distribuciones de GNU/Linux respetan completamente el estándar de los runlevels definido por la LSB, ya que algunos sistemas operativos utilizan diferentes niveles de ejecución o combinaciones de servicios y procesos. Sin embargo, la mayoría de las distribuciones respetan los niveles de ejecución definidos por el estándar de la LSB.

(f) Archivo /etc/inittab. ¿Cuál es su finalidad? ¿Qué tipo de información se almacena en él? ¿Cuál es la estructura de la información que en él se almacena?

El archivo "/etc/inittab" es un archivo de configuración que se encuentra en sistemas operativos basados en System V, como la mayoría de las distribuciones de GNU/Linux. Su finalidad es definir el comportamiento del sistema al iniciar y durante el proceso de ejecución. En particular, este archivo especifica el nivel de ejecución predeterminado, los procesos y servicios que se deben iniciar en cada nivel de ejecución y otras opciones de configuración del sistema.

En el archivo "/etc/inittab" se almacena información en forma de líneas de configuración. Cada línea de configuración define una tarea o servicio que debe realizarse durante el proceso de inicio. Estas líneas de configuración se componen de varios campos separados por dos puntos, incluyendo el identificador de tarea, el nivel de ejecución, la acción que se debe realizar y el comando o script que se debe ejecutar. Algunos de los campos más comunes son:

* Id: El identificador de la tarea o servicio que se va a ejecutar.
* Runlevel: El nivel de ejecución para el que se especifica la tarea o servicio.
* Action: La acción que se debe realizar, como iniciar o detener un servicio.
* Process: El comando o script que se debe ejecutar para realizar la tarea o servicio.

La estructura de la información almacenada en el archivo "/etc/inittab" es la siguiente:

identificador:tarea:runlevel:acción:comando

Donde cada campo está separado por el carácter ":" y tiene la siguiente descripción:

* identificador: Es un identificador único que se utiliza para hacer referencia a la tarea o servicio que se está configurando en la línea actual.
* tarea: Es una descripción breve de la tarea o servicio que se está configurando.
* runlevel: Es el nivel de ejecución en el que se debe ejecutar la tarea o servicio. Este campo puede tener los valores 0, 1, 2, 3, 4, 5 o 6, donde cada nivel de ejecución representa un estado del sistema con diferentes servicios y aplicaciones activas.
* acción: Es la acción que se debe realizar con la tarea o servicio en el nivel de ejecución especificado. Puede ser "respawn" para volver a ejecutar la tarea si se detiene inesperadamente, "wait" para esperar a que la tarea termine antes de continuar con otras tareas, o "once" para ejecutar la tarea solo una vez en el nivel de ejecución especificado.
* comando: Es el comando o script que se debe ejecutar para realizar la tarea o servicio. Puede ser una ruta absoluta o relativa a un archivo ejecutable o un script de shell.

(g) Suponga que se encuentra en el runlevel <X>. Indique qué comando(s) ejecutaría para cambiar al runlevel <Y>. ¿Este cambio es permanente? ¿Por qué?

Para cambiar del runlevel actual a otro, se puede utilizar el comando telinit seguido del número del runlevel al que se desea cambiar. Por ejemplo, si se encuentra en el runlevel 3 y se desea cambiar al runlevel 5, se debe ejecutar el siguiente comando con privilegios de superusuario:

sudo telinit 5

Este cambio no es permanente, ya que solo se realiza durante la sesión actual del sistema. Al reiniciar el sistema, se volverá al runlevel predeterminado configurado en el archivo "/etc/inittab". Si se desea cambiar el runlevel predeterminado, se debe editar el archivo "/etc/inittab" y modificar la línea correspondiente al runlevel predeterminado.

(h) Scripts RC. ¿Cuál es su finalidad? ¿Dónde se almacenan? Cuando un sistema GNU/Linux arranca o se detiene se ejecutan scripts, indique cómo determina qué script ejecutar ante cada acción. ¿Existe un orden para llamarlos? Justifique.

Los scripts RC (Run Commands) son scripts que se utilizan en sistemas GNU/Linux para automatizar la ejecución de tareas y servicios durante el proceso de inicio y apagado del sistema. La finalidad de estos scripts es realizar configuraciones y acciones necesarias para que el sistema pueda funcionar correctamente, como, por ejemplo, cargar módulos del kernel, configurar la red, iniciar servicios, etc.

Los scripts RC se almacenan en diferentes directorios dependiendo del sistema operativo y su distribución, pero generalmente se encuentran en el directorio /etc/rc.d o /etc/init.d. Cada script RC tiene un nombre que comienza con una letra que indica la secuencia de ejecución y seguido de un número que indica el orden relativo en el que se deben ejecutar.

Cuando un sistema GNU/Linux arranca o se detiene, se ejecutan los scripts RC correspondientes al runlevel actual. Cada script RC se ejecuta con un argumento que indica la acción a realizar, como "start" para iniciar un servicio o "stop" para detenerlo. El orden en que se ejecutan los scripts RC está determinado por el número que sigue a la letra inicial en el nombre del archivo. Por ejemplo, un script RC llamado "S10red" se ejecutará antes que un script llamado "S20apache".

El orden en que se llaman los scripts RC es importante, ya que algunos servicios o tareas dependen de otros para poder funcionar correctamente.

(i) ¿Qué es insserv? ¿Para qué se utiliza? ¿Qué ventajas provee respecto de un arranque tradicional?

insserv es una herramienta que se utiliza en sistemas basados en SystemV para administrar el orden de ejecución de los scripts RC durante el proceso de inicio y apagado del sistema. Esta herramienta lee los encabezados de los scripts RC y genera los enlaces simbólicos necesarios en los directorios de los runlevels correspondientes para que los scripts se ejecuten en el orden adecuado.

La ventaja principal que ofrece insserv respecto a un arranque tradicional es que simplifica el proceso de administración de los scripts RC. En un arranque tradicional, el administrador del sistema debe especificar manualmente el orden de ejecución de cada script RC modificando los nombres de los archivos. Esto puede resultar complicado y propenso a errores, especialmente en sistemas con múltiples scripts RC que tienen dependencias entre sí.

insserv automatiza este proceso y permite a los administradores del sistema configurar el orden de ejecución de los scripts RC de manera más eficiente y confiable. Además, permite a los administradores del sistema activar o desactivar scripts RC para ciertos runlevels con mayor facilidad, lo que puede simplificar la administración del sistema.

(j) ¿Cómo maneja Upstart el proceso de arranque del sistema?

Upstart es un sistema de inicio de servicios y demonios que se utiliza en algunas distribuciones de Linux, como Ubuntu, para gestionar el proceso de arranque del sistema. A diferencia del sistema SysV, que utiliza scripts de inicio y apagado en directorios específicos, Upstart utiliza archivos de configuración de servicio para definir los servicios y sus dependencias.

El proceso de arranque del sistema en Upstart consta de los siguientes pasos:

1. Carga del kernel: El kernel se carga en la memoria y se inicializan los drivers necesarios para el hardware del sistema.
2. Carga del initramfs: En algunos sistemas, se utiliza un archivo initramfs para cargar módulos adicionales y configurar el sistema antes de que se monte el sistema de archivos raíz.
3. Ejecución de Upstart: Upstart se ejecuta y carga los archivos de configuración de servicio desde el directorio /etc/init.
4. Arranque de servicios: Upstart arranca los servicios según su orden de inicio definido en los archivos de configuración y sus dependencias.
5. Llamada al runlevel: Upstart llama al runlevel especificado en el archivo /etc/inittab (si está presente), lo que inicia los servicios necesarios para ese nivel de ejecución.
6. Inicio de sesión: Una vez que se han iniciado todos los servicios necesarios, Upstart inicia el gestor de inicio de sesión (por ejemplo, LightDM o GDM), lo que permite que los usuarios inicien sesión en el sistema.

En resumen, Upstart maneja el proceso de arranque del sistema a través de archivos de configuración de servicio y sus dependencias, lo que permite un inicio más rápido y una gestión más flexible de los servicios.

(k) Cite las principales diferencias entre SystemV y Upstart.

Las principales diferencias entre SystemV y Upstart:

* Sistema de inicio: SystemV utiliza un sistema basado en scripts, mientras que Upstart utiliza archivos de configuración de servicio.
* Gestión de servicios: SystemV utiliza el comando "service" para iniciar, detener y reiniciar servicios, mientras que Upstart utiliza los comandos "start", "stop" y "restart".
* Dependencias de servicios: SystemV no maneja las dependencias de los servicios de manera eficiente, mientras que Upstart permite definir dependencias y secuencias de inicio de servicios.
* Control de errores: Upstart ofrece una mejor gestión de errores, permitiendo la recuperación de errores y reinicios automáticos.
* Gestión de eventos: Upstart permite que los servicios se inicien en respuesta a eventos específicos del sistema, como la conexión de un dispositivo USB o la creación de un archivo en un directorio específico.
* Escalabilidad: Upstart se considera más escalable que SystemV, ya que puede manejar un mayor número de servicios y procesos.

(l) Qué reemplaza a los scripts rc de SystemV en Upstart? ¿En que ubicación del filesystem se encuentran?

En Upstart, los scripts rc de SystemV son reemplazados por los archivos de configuración de servicio, que se encuentran en el directorio "/etc/init". Estos archivos tienen un formato diferente al de los scripts rc de SystemV y contienen información sobre cómo se debe iniciar, detener y reiniciar el servicio, así como las dependencias del servicio y otras configuraciones relevantes.

Cada archivo de configuración de servicio en Upstart tiene un nombre que coincide con el nombre del servicio que controla, seguido de la extensión ".conf". Por ejemplo, el archivo de configuración de servicio para el demonio de Apache en Upstart se llama "apache2.conf".

(m) Dado el siguiente job de upstart perteneciente al servicio de base de datos del mysql indique a qué hace referencia cada línea del mismo:

# MySQL Service

Description “MySQL Server”

Autor “info autor”

Start on (net-device-up

And local-filesystems

And runlevel [2345])

Stop on runlevel [016]

[…]

Exec /usr/sbin/mysqld

[…]

El job de Upstart que se presenta corresponde al servicio de base de datos MySQL, y cada línea se puede explicar de la siguiente manera:

* **# MySQL Service**: Este es un comentario que indica el nombre del servicio que se está definiendo en el archivo de configuración de Upstart.
* **Description "MySQL Server"**: Esta línea describe el servicio que se está definiendo. En este caso, el servicio es el servidor MySQL.
* **Author "info autor"**: Esta línea indica el nombre del autor o creador del archivo de configuración de Upstart.
* **Start on (net-device-up and local-filesystems and runlevel [2345])**: Esta línea especifica las condiciones que deben cumplirse para que el servicio se inicie automáticamente durante el proceso de arranque del sistema. En este caso, el servicio de MySQL se iniciará cuando se hayan activado los dispositivos de red (net-device-up), se hayan montado los sistemas de archivos locales (local-filesystems) y el sistema se encuentre en un nivel de ejecución de los especificados (runlevel [2345]).
* **Stop on runlevel [016]**: Esta línea especifica las condiciones que deben cumplirse para que el servicio se detenga automáticamente durante el proceso de apagado del sistema. En este caso, el servicio de MySQL se detendrá cuando el sistema cambie a un nivel de ejecución 0, 1 o 6 (runlevel [016]).
* **Exec /usr/sbin/mysqld**: Esta línea indica el comando o programa que se ejecutará cuando se inicie el servicio. En este caso, el comando /usr/sbin/mysqld inicia el demonio del servidor MySQL.

(n) ¿Qué es sytemd?

systemd es un sistema init y de administración de servicios para sistemas operativos basados en Linux. Fue diseñado como reemplazo del tradicional SystemV init, y proporciona una serie de mejoras y características adicionales.

Entre las características de systemd se incluyen:

* Arranque paralelo y optimizado de servicios, lo que reduce el tiempo de inicio del sistema.
* Control de dependencias y orden de arranque de servicios.
* Control de recursos y limitación de uso de recursos.
* Soporte para la activación y desactivación dinámica de servicios en tiempo de ejecución.
* Registro centralizado de mensajes de servicio y diagnóstico de problemas.
* Integración con el kernel de Linux para el control avanzado de características del sistema, como la gestión de cgroups y namespaces.
* Interfaz de línea de comandos (systemctl) para la gestión de servicios y unidades.

systemd está compuesto por un conjunto de demonios que se ejecutan en segundo plano en un sistema Linux y que proporcionan diversas funcionalidades para la gestión de servicios y el proceso de arranque del sistema.

Algunos de los demonios más importantes que componen systemd son:

* systemd: el proceso principal de systemd, que se ejecuta como PID 1 y es responsable de iniciar y coordinar todos los demás componentes de systemd.
* systemd-journald: el demonio de registro de systemd, que gestiona el registro de mensajes del sistema y de los servicios.
* systemd-logind: el demonio de gestión de sesiones de systemd, que se encarga de la gestión de sesiones de usuarios, dispositivos de entrada, etc.
* systemd-networkd: el demonio de gestión de redes de systemd, que proporciona herramientas para la configuración y gestión de redes.
* systemd-resolved: el demonio de resolución de DNS de systemd, que proporciona servicios de resolución de nombres de dominio y DNS.

En general, cada demonio de systemd está diseñado para proporcionar una funcionalidad específica y se ejecuta de forma independiente de los demás. Juntos, forman una plataforma completa y coherente para la gestión del sistema y los servicios en sistemas Linux.

systemd reemplazó a Upstart como el sistema de inicio predeterminado en muchas distribuciones de Linux a partir de 2015. Si bien ambos sistemas tienen algunas similitudes en su funcionalidad básica, systemd se ha convertido en el estándar de facto en la mayoría de las distribuciones de Linux modernas debido a su enfoque modular, su velocidad de inicio y su capacidad para proporcionar un mejor control y gestión de los servicios y procesos del sistema.

(ñ) ¿A qué hace referencia el concepto de activación de socket en systemd?

El concepto de activación de socket en systemd hace referencia a la capacidad del sistema para activar un servicio cuando se recibe una conexión a un socket específico. En otras palabras, systemd puede escuchar y esperar conexiones entrantes en un socket, y una vez que se recibe una conexión, puede iniciar el servicio correspondiente para manejar esa conexión.

Esta funcionalidad es útil para servicios que necesitan iniciar rápidamente cuando se recibe una conexión, como servidores web o bases de datos, ya que permite que el sistema inicie el servicio solo cuando es necesario y lo detenga cuando no hay conexiones activas. También puede mejorar el rendimiento del sistema, ya que los servicios solo se activan cuando se necesitan, en lugar de ejecutarse continuamente en segundo plano.

(o) ¿A qué hace referencia el concepto de cgroup?

El concepto de cgroup se refiere a la funcionalidad en el kernel de Linux que permite la gestión de recursos del sistema para procesos y grupos de procesos. "cgroup" es una abreviatura de "control group".

En términos simples, los cgroups son una forma de asignar y limitar los recursos del sistema, como la CPU, la memoria, la red y el almacenamiento, a grupos de procesos o a procesos individuales. De esta manera, se puede evitar que un proceso o grupo de procesos acapare todos los recursos del sistema, lo que podría afectar negativamente el rendimiento del sistema y la capacidad de otros procesos para ejecutarse de manera efectiva.

Además de limitar los recursos del sistema, los cgroups también permiten establecer prioridades y políticas de planificación de procesos, lo que puede ser útil para asegurar que los procesos críticos reciban los recursos que necesitan para funcionar correctamente.

3. Usuarios:

(a) ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?

En un sistema GNU/Linux, la información de los usuarios se guarda principalmente en los siguientes archivos:

* /etc/passwd: este archivo contiene información sobre cada usuario, como el nombre de usuario, la ubicación de su directorio de inicio, el identificador de usuario (UID), el identificador de grupo primario (GID) y la shell predeterminada.
* /etc/shadow: este archivo contiene las contraseñas cifradas de los usuarios y la información relacionada con la caducidad de las contraseñas.
* /etc/group: este archivo contiene información sobre cada grupo, como el nombre del grupo y el identificador de grupo (GID), así como una lista de los usuarios que pertenecen al grupo.

(b) ¿A qué hacen referencia las siglas UID y GID? ¿Pueden coexistir UIDs iguales en un sistema GNU/Linux? Justifique

UID significa "User ID" y se refiere a un identificador numérico único asignado a cada usuario en un sistema GNU/Linux. GID significa "Group ID" y se refiere a un identificador numérico único asignado a cada grupo en un sistema GNU/Linux.

En un sistema GNU/Linux, no puede haber dos usuarios con el mismo UID. Esto se debe a que el UID se utiliza para identificar de manera única a un usuario en el sistema. Sin embargo, puede haber varios usuarios en diferentes sistemas que tienen el mismo UID, ya que los UIDs se asignan localmente en cada sistema y no están globalmente coordinados.

En cuanto a los GIDs, puede haber varios grupos con el mismo GID, ya que el GID se utiliza para identificar de manera única a un grupo en el sistema, pero no a los usuarios individuales.

(c) ¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la UID del *root*?

En sistemas GNU/Linux, el usuario root es el usuario con permisos de superusuario, lo que significa que tiene acceso completo y sin restricciones al sistema. El usuario root tiene la capacidad de realizar cualquier acción en el sistema, incluyendo la instalación y eliminación de software, la configuración de redes, el acceso a archivos protegidos y la creación de otros usuarios y grupos.

Por diseño, existe solo un usuario root en un sistema GNU/Linux. La UID del usuario root es siempre 0.

(d) Agregue un nuevo usuario llamado iso2017 a su instalación de GNU/Linux, especifique que su home sea creada en /home/iso\_2017, y hágalo miembro del grupo catedra (si no existe, deberá crearlo). Luego, sin iniciar sesión como este usuario cree un archivo en su home personal que le pertenezca. Luego de todo esto, borre el usuario y verifique que no queden registros de él en los archivos de información de los usuarios y grupos

Para agregar un nuevo usuario llamado iso2017 en GNU/Linux y especificar su directorio home y grupo, se deben seguir los siguientes pasos:

1. Crear el grupo "catedra" si no existe:

sudo groupadd catedra

1. Crear el usuario "iso2017" con el comando adduser, especificando su directorio home y grupo:

sudo adduser iso2017 --home /home/iso\_2017 --ingroup catedra

1. Crear un archivo en el directorio home del usuario "iso2017":

sudo -u iso2017 touch /home/iso\_2017/archivo\_prueba.txt

1. Verificar que el archivo se haya creado:

ls -l /home/iso\_2017/

1. Eliminar al usuario "iso2017" junto con su directorio home:

sudo deluser iso2017 --remove-home

1. Verificar que el usuario y su directorio home hayan sido eliminados correctamente:

cat /etc/passwd | grep iso2017 # no debe aparecer ningún resultado

cat /etc/group | grep catedra # debe aparecer el grupo "catedra" pero sin la entrada del usuario "iso2017"

(e) Investigue la funcionalidad y parámetros de los siguientes comandos:

* useradd ó adduser
* usermod
* userdel
* su
* groupadd
* who
* groupdel
* passwd

4. FileSystem:

(a) ¿Cómo son definidos los permisos sobre archivos en un sistema GNU/Linux?

(b) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con los permisos en GNU/Linux:

* chmod
* chown
* chgrp

(c) Al utilizar el comando chmod generalmente se utiliza una notación octal asociada para definir permisos. ¿Qué significa esto? ¿A qué hace referencia cada valor?

(d) ¿Existe la posibilidad de que algún usuario del sistema pueda acceder a determinado archivo para el cual no posee permisos? Nombrelo, y realice las pruebas correspondientes

(e) Explique los conceptos de “full path name” y “relative path name”. De ejemplos claros de cada uno de ellos.

(f) ¿Con qué comando puede determinar en qué directorio se encuentra actualmente? ¿Existe alguna forma de ingresar a su directorio personal sin necesidad de escribir todo el path completo? ¿Podría utilizar la misma idea para acceder a otros directorios? ¿Cómo? Explique con un ejemplo

(g) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso del FileSystem:

* cd
* umount
* mkdir
* du
* rmdir
* df
* mount
* ln
* ls
* pwd
* cp
* mv

5. Procesos:

(a) ¿Qué es un proceso? ¿A qué hacen referencia las siglas *PID y PPID*? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.

Un proceso es una instancia de un programa en ejecución en un sistema operativo. Cada proceso tiene su propio espacio de memoria, identidad de usuario y grupo, estado y contexto de ejecución.

Las siglas PID significan "Process ID" (ID de Proceso) y se refiere a un número único que se le asigna a cada proceso en ejecución en el sistema operativo para su identificación. El PPID (Parent Process ID) es el PID del proceso padre que inició el proceso en cuestión.

En general, todos los procesos en GNU/Linux tienen estos atributos, sin embargo, hay algunas excepciones. Por ejemplo, algunos procesos especiales del kernel pueden no tener un PID o pueden tener un PID igual a 0.

Además del PID y el PPID, los procesos en GNU/Linux tienen otros atributos como el estado de ejecución (en ejecución, suspendido, detenido, etc.), la prioridad de ejecución, los recursos asignados (memoria, CPU, etc.), los archivos abiertos, los puertos de red utilizados y la línea de comandos utilizada para iniciar el proceso, entre otros. Estos atributos se pueden ver y administrar a través de herramientas como el comando ps o el administrador de tareas del entorno gráfico.

(b) Indique qué comandos se podrían utilizar para ver qué procesos están en ejecución en un sistema GNU/Linux.

Existen varios comandos para ver qué procesos están en ejecución en un sistema GNU/Linux. Algunos son:

* **ps**: Muestra información detallada sobre los procesos en ejecución. Por ejemplo, ps **aux** muestra todos los procesos en ejecución junto con sus detalles.
* top: Muestra en tiempo real los procesos en ejecución, sus estadísticas y el uso de recursos.
* **htop**: Similar a top, muestra los procesos en ejecución junto con su uso de recursos, pero con una interfaz más interactiva y fácil de usar.
* **pgrep**: Busca procesos por nombre o por otros atributos y muestra sus PIDs.
* **pidof**: Busca el PID de un proceso a partir de su nombre.
* **systemctl**: Muestra el estado de los servicios y procesos de systemd.

(c) ¿Qué significa que un proceso se está ejecutando en Background? ¿Y en Foreground?

Cuando un proceso se está ejecutando en background, significa que está corriendo en segundo plano, es decir, que no está en primer plano y no tiene el control directo de la consola o terminal. Esto significa que se pueden ejecutar otros comandos y programas mientras el proceso en background continúa corriendo.

En cambio, cuando un proceso se está ejecutando en foreground, significa que está en primer plano y tiene el control directo de la consola o terminal, es decir, que cualquier entrada de teclado y salida de pantalla se realiza a través del proceso. En este caso, no se pueden ejecutar otros comandos y programas hasta que el proceso en primer plano finalice o se suspenda temporalmente con un comando como Ctrl + Z.

Cuando un proceso se ejecuta en foreground, significa que se está ejecutando en primer plano y que está interactuando directamente con el usuario a través de la terminal. En otras palabras, el proceso está en una sesión de terminal que está activa en la pantalla del usuario y el usuario puede enviar comandos directamente al proceso y recibir respuestas de inmediato.

Por otro lado, cuando un proceso se ejecuta en background, significa que está ejecutándose en segundo plano y que no está interactuando directamente con el usuario a través de la terminal. En este caso, el proceso se ejecuta en una sesión de terminal separada y está desvinculado de la sesión de terminal actual del usuario. Esto significa que el usuario puede continuar trabajando en la terminal actual y ejecutar otros comandos mientras el proceso se ejecuta en segundo plano.

Para ejecutar un proceso en segundo plano en GNU/Linux se puede agregar el símbolo "&" al final del comando en la terminal. Por ejemplo, si queremos ejecutar el comando "sleep 10" en segundo plano, se podría escribir "sleep 10 &". De esta forma, el proceso se ejecutará en segundo plano y se liberará la terminal para poder seguir trabajando.

(d) ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Como puedo hacer para pasar un proceso de background a foreground y viceversa?

Para ejecutar un proceso en Background se puede utilizar el operador "&" al final del comando en la terminal.

Para pasar un proceso de background a foreground, se puede utilizar el comando "fg", seguido del número del job que se desea recuperar. Los jobs son procesos en segundo plano que han sido enviados al fondo con el operador "&".

Para ver la lista de los jobs que están en segundo plano, se puede utilizar el comando "jobs". Este comando muestra una lista numerada de los procesos en segundo plano que se están ejecutando. Cada proceso tendrá asignado un número de job que se utilizará para referirse a él.

Para enviar un proceso de foreground a background se puede utilizar el comando "ctrl+z" en la terminal para detener el proceso y luego enviarlo al background con el comando "bg".

(e) Pipe ( | ). ¿Cuál es su finalidad? Cite ejemplos de su utilización.

El operador pipe (|) en GNU/Linux se utiliza para conectar la salida de un proceso con la entrada de otro proceso, permitiendo que los procesos trabajen juntos en una secuencia de comandos. La finalidad principal del operador pipe es la de permitir la comunicación entre procesos de forma sencilla y eficiente.

Por ejemplo, si se quiere buscar un archivo en el directorio actual y ver si contiene una cadena específica de texto, se puede utilizar el siguiente comando en la línea de comandos:

ls | grep "texto"

En este ejemplo, el comando "ls" muestra el contenido del directorio actual y el comando "grep" busca el texto especificado en la salida de "ls". El operador pipe se encarga de enviar la salida de "ls" a la entrada de "grep".

Otro ejemplo podría ser el siguiente:

ps aux | grep "proceso"

Este comando muestra la lista de procesos que están siendo ejecutados actualmente en el sistema, y luego filtra la lista para mostrar solo aquellos que contengan la palabra "proceso".

En general, el operador pipe es muy útil para combinar comandos y realizar tareas complejas utilizando una secuencia de comandos simples y sencillos.

(f) Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad? Cite ejemplos de utilización.

La redirección en GNU/Linux permite manejar la entrada y salida de datos de los comandos de manera eficiente y flexible. Existen varios tipos de redirecciones:

* *Redirección de salida (>)*: se utiliza para enviar la salida de un comando a un archivo o dispositivo en lugar de a la pantalla. Por ejemplo, para guardar la salida del comando ls en un archivo llamado lista.txt, se podría utilizar el comando: *ls > lista.txt*. Si el archivo ya existe, se sobrescribirá. Si se desea añadir la salida al final del archivo sin borrar su contenido anterior, se puede utilizar >> en lugar de >.
* *Redirección de entrada (<)*: se utiliza para enviar la entrada de un comando desde un archivo en lugar de desde el teclado. Por ejemplo, para enviar el contenido del archivo archivo.txt al comando cat, se podría utilizar el comando: cat < archivo.txt.
* *Redirección de errores (2>)*: se utiliza para enviar los mensajes de error de un comando a un archivo o dispositivo en lugar de a la pantalla. Por ejemplo, para guardar los mensajes de error del comando ls en un archivo llamado errores.txt, se podría utilizar el comando: ls /directorio/inexistente 2> errores.txt. Si se desea redirigir tanto la salida como los errores a un mismo archivo, se puede utilizar &> en lugar de >.
* *Redirección de entrada/salida (<>)*: se utiliza para enviar tanto la entrada como la salida de un comando desde y hacia un archivo en lugar de desde y hacia el teclado y la pantalla, respectivamente.
* *Redirección de pipes (|)*: se utiliza para enviar la salida de un comando como entrada de otro comando. Por ejemplo, para buscar la palabra "hola" en el archivo archivo.txt y mostrar sólo las líneas que la contengan, se podría utilizar el comando: grep hola archivo.txt | less.

(g) Comando *kill*. ¿Cuál es su funcionalidad? Cite ejemplos.

El comando kill es utilizado en sistemas operativos tipo Unix/Linux para enviar señales a procesos en ejecución, permitiendo controlar su comportamiento. La señal por defecto enviada por kill es la señal TERM (15), que indica al proceso que debe terminar de manera ordenada.

A continuación se muestran algunos ejemplos de cómo utilizar el comando kill:

* Matar un proceso con PID específico:

$ kill PID

Donde PID es el número de identificación del proceso que se desea matar.

* Enviar una señal específica a un proceso:

$ kill -SIGNAL PID

Donde SIGNAL es el número o el nombre de la señal que se desea enviar al proceso.

* Matar todos los procesos que pertenecen a un usuario específico:

$ pkill -u username

Donde username es el nombre del usuario cuyos procesos se desean matar.

* Matar todos los procesos con un nombre específico:

$ pkill nombre\_proceso

Donde nombre\_proceso es el nombre del proceso que se desea matar.

Es importante destacar que el uso del comando kill debe realizarse con precaución, ya que puede provocar la pérdida de datos o dañar el sistema si se mata un proceso crítico o importante para el funcionamiento del sistema.

(h) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el manejo de procesos en GNU/Linux. Además, compárelos entre ellos:

* **ps**: El comando ps muestra información sobre los procesos en ejecución en el sistema. Algunos de los parámetros más utilizados son:
  + **-a**: muestra los procesos de todos los usuarios.
  + **-u**: muestra los procesos pertenecientes a un usuario específico.
  + **-x**: muestra los procesos que no están asociados con una terminal.
  + **aux**: muestra información detallada de los procesos.
* **kill**: El comando kill envía una señal a un proceso especificado. Algunos de los parámetros más utilizados son:
  + **-s <señal>**: especifica la señal a enviar al proceso.
  + **-9**: envía una señal SIGKILL al proceso, lo que lo detiene de manera inmediata sin permitir que realice ninguna acción adicional.
* **pstree**: El comando pstree muestra los procesos del sistema en forma de árbol, con los procesos secundarios agrupados bajo sus procesos padres. Algunos de los parámetros más utilizados son:
  + **-p**: muestra los identificadores de proceso (PIDs) en lugar de los nombres de los procesos.
  + **-u**: muestra los nombres de usuario de los procesos.
* **killall**: El comando killall envía una señal a todos los procesos que coincidan con un nombre de proceso especificado. Algunos de los parámetros más utilizados son:
  + **-s <señal>**: especifica la señal a enviar a los procesos.
* **top**: El comando top muestra información en tiempo real sobre los procesos en ejecución en el sistema, incluyendo la cantidad de CPU y memoria que están utilizando. Algunos de los parámetros más utilizados son:
  + **-d <segundos>**: especifica el intervalo de actualización de la pantalla.
  + **-u <usuario>**: muestra solo los procesos pertenecientes al usuario especificado.
* **nice**: El comando nice cambia la prioridad de un proceso. Algunos de los parámetros más utilizados son:
  + **-n <valor>**: especifica el valor de la prioridad del proceso, donde los valores más bajos corresponden a una prioridad más alta.

Estos comandos tienen diferentes funcionalidades y se utilizan para diferentes tareas relacionadas con el manejo de procesos en GNU/Linux. Por ejemplo, **ps** se utiliza para ver información detallada de los procesos, **kill** para enviar señales a procesos específicos, **pstree** para visualizar procesos en forma de árbol, **killall** para enviar señales a procesos que coincidan con un nombre de proceso específico, **top** para monitorear la actividad del sistema en tiempo real, y **nice** para cambiar la prioridad de un proceso.

6. Otros comandos de Linux (Indique funcionalidad y parámetros):

(a) ¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?

El concepto de empaquetar archivos en GNU/Linux hace referencia a la acción de crear un archivo que contenga uno o varios archivos, con el fin de facilitar su transporte, almacenamiento o distribución.

Los formatos de archivo más utilizados para empaquetar en GNU/Linux son:

* **TAR**: es un formato de archivo que permite empaquetar uno o varios archivos en un solo archivo, manteniendo la estructura de directorios. La sintaxis básica del comando tar es:

tar -opciones archivo\_destino archivos\_a\_empaquetar

Algunas opciones comunes de tar son:

* + -c: crear un archivo tar
  + -x: extraer archivos de un archivo tar
  + -v: mostrar los archivos que se están procesando
  + -f: especificar el nombre del archivo tar
* **GZIP**: es un formato de compresión de archivos que se utiliza junto con el formato tar para empaquetar y comprimir uno o varios archivos. La sintaxis básica del comando gzip es:

gzip archivo\_a\_comprimir

Para descomprimir un archivo en formato gzip, se utiliza el comando gunzip:

gunzip archivo\_comprimido.gz

* **ZIP**: es un formato de archivo comprimido muy utilizado en Windows, pero también es compatible con GNU/Linux. La sintaxis básica del comando zip es:

zip archivo\_destino archivos\_a\_empaquetar

Algunas opciones comunes de zip son:

* + -r: empaquetar directorios de forma recursiva
  + -v: mostrar los archivos que se están procesando
  + -q: ejecutar zip en modo silencioso
* **RAR**: es otro formato de archivo comprimido muy utilizado en Windows, pero también es compatible con GNU/Linux. Para trabajar con archivos RAR en GNU/Linux es necesario instalar el programa unrar. La sintaxis básica del comando unrar es:

unrar opciones archivo\_a\_descomprimir destino

Algunas opciones comunes de unrar son:

* + e: extraer archivos sin crear directorios
  + x: extraer archivos y directorios
  + v: mostrar el progreso de la operación

(b) Seleccione 4 archivos dentro de algún directorio al que tenga permiso y sume el tamaño de cada uno de estos archivos. Cree un archivo empaquetado conteniendo estos 4 archivos y compare los tamaños de los mismos. ¿Qué característica nota?

Puedo sumar los tamaños de los 4 archivos que deseo empacar utilizando el comando **du**:

du -h archivo1 archivo2 archivo3 archivo4

Para crear un archivo empaquetado con estos archivos, puedo utilizar el comando **tar** con la opción **cvzf** para crear un archivo .tar.gz:

tar cvzf archivos.tar.gz archivo1 archivo2 archivo3 archivo4

Al usar el comando tar sin especificar las opciones cvzf, se creará un archivo tar que contendrá los archivos seleccionados, pero no estará comprimido. Al usar la opción cvzf, se creará un archivo tar comprimido utilizando la compresión gzip.

Una vez creado el archivo empaquetado, puedo comparar su tamaño con la suma de los tamaños de los 4 archivos originales utilizando el comando **du**:

du -h archivos.tar.gz

En general, es posible que el tamaño del archivo empaquetado sea ligeramente menor que la suma de los tamaños de los archivos originales, debido a la compresión que se aplica en el proceso de empaquetado. Sin embargo, la diferencia en tamaño dependerá del tipo de archivo y del nivel de compresión utilizado.

(c) ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo? Indique la secuencia de comandos ejecutados.

Para comprimir 4 archivos en uno solo podemos utilizar el comando tar. La secuencia de comandos a ejecutar sería la siguiente:

1. Acceder al directorio donde se encuentran los archivos que se desean comprimir:

cd /ruta/del/directorio/

1. Comprimir los archivos en un archivo tar utilizando el comando **tar**, utilizando la opción **-cf** para crear un archivo y **-v** para mostrar el progreso, y especificando el nombre del archivo que se va a crear y los archivos que se van a comprimir:

tar -cvf archivos.tar archivo1 archivo2 archivo3 archivo4

1. Comprimir el archivo tar utilizando el comando **gzip**, utilizando la opción **-v** para mostrar el progreso y especificando el nombre del archivo que se va a comprimir:

gzip -v archivos.tar

Una vez ejecutados estos comandos, se habrá creado un archivo comprimido llamado archivos.tar.gz en el directorio actual, que contendrá los 4 archivos originales comprimidos en uno solo.

(d) ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?

Sí, se puede comprimir un conjunto de archivos utilizando un único comando. Utilizando el comando tar para empaquetar los archivos y luego comprimir el archivo resultante utilizando alguna herramienta de compresión como gzip o bzip2.

Por ejemplo, para comprimir todos los archivos de la carpeta documentos en un archivo documentos.tar.gz, se puede utilizar el siguiente comando:

tar -czvf documentos.tar.gz documentos/

Donde:

* **c**: crea un nuevo archivo tar
* **z**: comprime el archivo tar utilizando gzip
* **v**: muestra el progreso del empaquetamiento y compresión
* **f**: especifica el nombre del archivo de salida

De esta forma, se obtendrá un archivo documentos.tar.gz que contiene todos los archivos de la carpeta documentos.

(e) Investigue la funcionalidad de los siguientes comandos:

* tar: el comando tar es utilizado para crear, manipular y extraer archivos comprimidos en formato tar. Este comando es muy común en sistemas Unix y se utiliza para hacer copias de seguridad, transferir archivos y mantener una estructura organizada de archivos y directorios. Algunos de los parámetros más utilizados son -c para crear un archivo tar, -x para extraer un archivo tar y -f para especificar el nombre del archivo tar.
* grep: el comando grep es una herramienta de búsqueda que permite buscar cadenas de texto dentro de archivos o directorios. Su funcionalidad se basa en la utilización de expresiones regulares para encontrar patrones de texto específicos. Algunos de los parámetros más utilizados son -i para realizar una búsqueda insensible a mayúsculas y minúsculas, -v para buscar líneas que no contengan la cadena de texto especificada y -r para buscar en un directorio y todos sus subdirectorios.
* gzip: el comando gzip es utilizado para comprimir archivos en formato GZIP. Este formato de compresión es muy común en sistemas Unix y Linux, y es utilizado para reducir el tamaño de archivos y ahorrar espacio en disco. Algunos de los parámetros más utilizados son -c para escribir la salida comprimida en la salida estándar, -d para descomprimir un archivo GZIP y -r para comprimir todos los archivos de un directorio y sus subdirectorios.
* zgrep: el comando zgrep es similar al comando grep, pero está diseñado para buscar cadenas de texto dentro de archivos comprimidos en formato GZIP. Algunos de los parámetros más utilizados son -i para realizar una búsqueda insensible a mayúsculas y minúsculas y -v para buscar líneas que no contengan la cadena de texto especificada.
* wc: el comando wc (word count) se utiliza para contar el número de líneas, palabras y caracteres de un archivo. Algunos de los parámetros más utilizados son -l para contar el número de líneas, -w para contar el número de palabras y -c para contar el número de caracteres. También es posible utilizar varios parámetros en conjunto para obtener información detallada sobre el archivo.

7. Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando, indique la razón:

* ls −l > prueba: Genera un listado detallado de los archivos y directorios del directorio actual y lo redirige a un archivo llamado "prueba".
* ps > PRUEBA
* chmod 710 prueba
* chown root:root PRUEBA
* chmod 777 PRUEBA
* chmod 700 / etc / passwd
* passwd root
* rm PRUEBA
* man / etc / shadow
* find / −name ∗.conf
* usermod root −d /home/ newroot −L
* cd /root
* rm ∗
* cd / etc
* cp ∗ /home −R
* shutdown